

GiNaC

1.8.10

Generated by Doxygen 1.16.1



<b>1 Directory Hierarchy</b>	<b>1</b>
1.1 Directories	1
<b>2 Namespace Index</b>	<b>5</b>
2.1 Namespace List	5
<b>3 Hierarchical Index</b>	<b>7</b>
3.1 Class Hierarchy	7
<b>4 Class Index</b>	<b>11</b>
4.1 Class List	11
<b>5 File Index</b>	<b>19</b>
5.1 File List	19
<b>6 Directory Documentation</b>	<b>23</b>
6.1 ginac Directory Reference	23
<b>7 Namespace Documentation</b>	<b>29</b>
7.1 GiNaC Namespace Reference	29
7.1.1 Typedef Documentation	68
7.1.1.1 archive_node_id	68
7.1.1.2 archive_atom	68
7.1.1.3 synthesise_func	68
7.1.1.4 unarchive_map_t	69
7.1.1.5 exvector	69
7.1.1.6 exset	69
7.1.1.7 exmap	69
7.1.1.8 evalffunctype	69
7.1.1.9 FUNCP_1P	69
7.1.1.10 FUNCP_2P	69
7.1.1.11 FUNCP_CUBA	69
7.1.1.12 epvector	69
7.1.1.13 epp	70
7.1.1.14 exprseq	70
7.1.1.15 paramset	70
7.1.1.16 eval_funcp	70
7.1.1.17 evalf_funcp	70
7.1.1.18 conjugate_funcp	70
7.1.1.19 real_part_funcp	70
7.1.1.20 imag_part_funcp	70
7.1.1.21 expand_funcp	70
7.1.1.22 derivative_funcp	70
7.1.1.23 expl_derivative_funcp	71

7.1.1.24 power_funcp . . . . .	71
7.1.1.25 series_funcp . . . . .	71
7.1.1.26 print_funcp . . . . .	71
7.1.1.27 info_funcp . . . . .	71
7.1.1.28 eval_funcp_1 . . . . .	71
7.1.1.29 evalf_funcp_1 . . . . .	71
7.1.1.30 conjugate_funcp_1 . . . . .	71
7.1.1.31 real_part_funcp_1 . . . . .	71
7.1.1.32 imag_part_funcp_1 . . . . .	71
7.1.1.33 expand_funcp_1 . . . . .	72
7.1.1.34 derivative_funcp_1 . . . . .	72
7.1.1.35 expl_derivative_funcp_1 . . . . .	72
7.1.1.36 power_funcp_1 . . . . .	72
7.1.1.37 series_funcp_1 . . . . .	72
7.1.1.38 print_funcp_1 . . . . .	72
7.1.1.39 info_funcp_1 . . . . .	72
7.1.1.40 eval_funcp_2 . . . . .	72
7.1.1.41 evalf_funcp_2 . . . . .	72
7.1.1.42 conjugate_funcp_2 . . . . .	72
7.1.1.43 real_part_funcp_2 . . . . .	73
7.1.1.44 imag_part_funcp_2 . . . . .	73
7.1.1.45 expand_funcp_2 . . . . .	73
7.1.1.46 derivative_funcp_2 . . . . .	73
7.1.1.47 expl_derivative_funcp_2 . . . . .	73
7.1.1.48 power_funcp_2 . . . . .	73
7.1.1.49 series_funcp_2 . . . . .	73
7.1.1.50 print_funcp_2 . . . . .	73
7.1.1.51 info_funcp_2 . . . . .	73
7.1.1.52 eval_funcp_3 . . . . .	73
7.1.1.53 evalf_funcp_3 . . . . .	74
7.1.1.54 conjugate_funcp_3 . . . . .	74
7.1.1.55 real_part_funcp_3 . . . . .	74
7.1.1.56 imag_part_funcp_3 . . . . .	74
7.1.1.57 expand_funcp_3 . . . . .	74
7.1.1.58 derivative_funcp_3 . . . . .	74
7.1.1.59 expl_derivative_funcp_3 . . . . .	74
7.1.1.60 power_funcp_3 . . . . .	74
7.1.1.61 series_funcp_3 . . . . .	74
7.1.1.62 print_funcp_3 . . . . .	74
7.1.1.63 info_funcp_3 . . . . .	75
7.1.1.64 eval_funcp_4 . . . . .	75
7.1.1.65 evalf_funcp_4 . . . . .	75

---

7.1.1.66 conjugate_funcp_4 . . . . .	75
7.1.1.67 real_part_funcp_4 . . . . .	75
7.1.1.68 imag_part_funcp_4 . . . . .	75
7.1.1.69 expand_funcp_4 . . . . .	75
7.1.1.70 derivative_funcp_4 . . . . .	75
7.1.1.71 expl_derivative_funcp_4 . . . . .	75
7.1.1.72 power_funcp_4 . . . . .	75
7.1.1.73 series_funcp_4 . . . . .	76
7.1.1.74 print_funcp_4 . . . . .	76
7.1.1.75 info_funcp_4 . . . . .	76
7.1.1.76 eval_funcp_5 . . . . .	76
7.1.1.77 evalf_funcp_5 . . . . .	76
7.1.1.78 conjugate_funcp_5 . . . . .	76
7.1.1.79 real_part_funcp_5 . . . . .	76
7.1.1.80 imag_part_funcp_5 . . . . .	76
7.1.1.81 expand_funcp_5 . . . . .	76
7.1.1.82 derivative_funcp_5 . . . . .	77
7.1.1.83 expl_derivative_funcp_5 . . . . .	77
7.1.1.84 power_funcp_5 . . . . .	77
7.1.1.85 series_funcp_5 . . . . .	77
7.1.1.86 print_funcp_5 . . . . .	77
7.1.1.87 info_funcp_5 . . . . .	77
7.1.1.88 eval_funcp_6 . . . . .	77
7.1.1.89 evalf_funcp_6 . . . . .	77
7.1.1.90 conjugate_funcp_6 . . . . .	77
7.1.1.91 real_part_funcp_6 . . . . .	78
7.1.1.92 imag_part_funcp_6 . . . . .	78
7.1.1.93 expand_funcp_6 . . . . .	78
7.1.1.94 derivative_funcp_6 . . . . .	78
7.1.1.95 expl_derivative_funcp_6 . . . . .	78
7.1.1.96 power_funcp_6 . . . . .	78
7.1.1.97 series_funcp_6 . . . . .	78
7.1.1.98 print_funcp_6 . . . . .	78
7.1.1.99 info_funcp_6 . . . . .	78
7.1.1.100 eval_funcp_7 . . . . .	79
7.1.1.101 evalf_funcp_7 . . . . .	79
7.1.1.102 conjugate_funcp_7 . . . . .	79
7.1.1.103 real_part_funcp_7 . . . . .	79
7.1.1.104 imag_part_funcp_7 . . . . .	79
7.1.1.105 expand_funcp_7 . . . . .	79
7.1.1.106 derivative_funcp_7 . . . . .	79
7.1.1.107 expl_derivative_funcp_7 . . . . .	79

7.1.1.108 power_funcp_7 . . . . .	79
7.1.1.109 series_funcp_7 . . . . .	80
7.1.1.110 print_funcp_7 . . . . .	80
7.1.1.111 info_funcp_7 . . . . .	80
7.1.1.112 eval_funcp_8 . . . . .	80
7.1.1.113 evalf_funcp_8 . . . . .	80
7.1.1.114 conjugate_funcp_8 . . . . .	80
7.1.1.115 real_part_funcp_8 . . . . .	80
7.1.1.116 imag_part_funcp_8 . . . . .	80
7.1.1.117 expand_funcp_8 . . . . .	80
7.1.1.118 derivative_funcp_8 . . . . .	81
7.1.1.119 expl_derivative_funcp_8 . . . . .	81
7.1.1.120 power_funcp_8 . . . . .	81
7.1.1.121 series_funcp_8 . . . . .	81
7.1.1.122 print_funcp_8 . . . . .	81
7.1.1.123 info_funcp_8 . . . . .	81
7.1.1.124 eval_funcp_9 . . . . .	81
7.1.1.125 evalf_funcp_9 . . . . .	81
7.1.1.126 conjugate_funcp_9 . . . . .	81
7.1.1.127 real_part_funcp_9 . . . . .	82
7.1.1.128 imag_part_funcp_9 . . . . .	82
7.1.1.129 expand_funcp_9 . . . . .	82
7.1.1.130 derivative_funcp_9 . . . . .	82
7.1.1.131 expl_derivative_funcp_9 . . . . .	82
7.1.1.132 power_funcp_9 . . . . .	82
7.1.1.133 series_funcp_9 . . . . .	82
7.1.1.134 print_funcp_9 . . . . .	82
7.1.1.135 info_funcp_9 . . . . .	82
7.1.1.136 eval_funcp_10 . . . . .	83
7.1.1.137 evalf_funcp_10 . . . . .	83
7.1.1.138 conjugate_funcp_10 . . . . .	83
7.1.1.139 real_part_funcp_10 . . . . .	83
7.1.1.140 imag_part_funcp_10 . . . . .	83
7.1.1.141 expand_funcp_10 . . . . .	83
7.1.1.142 derivative_funcp_10 . . . . .	83
7.1.1.143 expl_derivative_funcp_10 . . . . .	83
7.1.1.144 power_funcp_10 . . . . .	83
7.1.1.145 series_funcp_10 . . . . .	84
7.1.1.146 print_funcp_10 . . . . .	84
7.1.1.147 info_funcp_10 . . . . .	84
7.1.1.148 eval_funcp_11 . . . . .	84
7.1.1.149 evalf_funcp_11 . . . . .	84

7.1.1.150 conjugate_funcp_11	84
7.1.1.151 real_part_funcp_11	84
7.1.1.152 imag_part_funcp_11	84
7.1.1.153 expand_funcp_11	84
7.1.1.154 derivative_funcp_11	85
7.1.1.155 expl_derivative_funcp_11	85
7.1.1.156 power_funcp_11	85
7.1.1.157 series_funcp_11	85
7.1.1.158 print_funcp_11	85
7.1.1.159 info_funcp_11	85
7.1.1.160 eval_funcp_12	85
7.1.1.161 evalf_funcp_12	85
7.1.1.162 conjugate_funcp_12	86
7.1.1.163 real_part_funcp_12	86
7.1.1.164 imag_part_funcp_12	86
7.1.1.165 expand_funcp_12	86
7.1.1.166 derivative_funcp_12	86
7.1.1.167 expl_derivative_funcp_12	86
7.1.1.168 power_funcp_12	86
7.1.1.169 series_funcp_12	86
7.1.1.170 print_funcp_12	87
7.1.1.171 info_funcp_12	87
7.1.1.172 eval_funcp_13	87
7.1.1.173 evalf_funcp_13	87
7.1.1.174 conjugate_funcp_13	87
7.1.1.175 real_part_funcp_13	87
7.1.1.176 imag_part_funcp_13	87
7.1.1.177 expand_funcp_13	87
7.1.1.178 derivative_funcp_13	88
7.1.1.179 expl_derivative_funcp_13	88
7.1.1.180 power_funcp_13	88
7.1.1.181 series_funcp_13	88
7.1.1.182 print_funcp_13	88
7.1.1.183 info_funcp_13	88
7.1.1.184 eval_funcp_14	88
7.1.1.185 evalf_funcp_14	88
7.1.1.186 conjugate_funcp_14	89
7.1.1.187 real_part_funcp_14	89
7.1.1.188 imag_part_funcp_14	89
7.1.1.189 expand_funcp_14	89
7.1.1.190 derivative_funcp_14	89
7.1.1.191 expl_derivative_funcp_14	89

7.1.1.192 power_funcp_14	89
7.1.1.193 series_funcp_14	89
7.1.1.194 print_funcp_14	90
7.1.1.195 info_funcp_14	90
7.1.1.196 eval_funcp_exvector	90
7.1.1.197 evalf_funcp_exvector	90
7.1.1.198 conjugate_funcp_exvector	90
7.1.1.199 real_part_funcp_exvector	90
7.1.1.200 imag_part_funcp_exvector	90
7.1.1.201 expand_funcp_exvector	90
7.1.1.202 derivative_funcp_exvector	90
7.1.1.203 expl_derivative_funcp_exvector	90
7.1.1.204 power_funcp_exvector	91
7.1.1.205 series_funcp_exvector	91
7.1.1.206 print_funcp_exvector	91
7.1.1.207 info_funcp_exvector	91
7.1.1.208 exhashmap	91
7.1.1.209 spmap	91
7.1.1.210 lookup_map	91
7.1.1.211 lst	91
7.1.1.212 uintvector	91
7.1.1.213 unsignedvector	91
7.1.1.214 exvectorvector	92
7.1.1.215 sym_desc_vec	92
7.1.1.216 digits_changed_callback	92
7.1.1.217 print_context_class_info	92
7.1.1.218 registered_class_info	92
7.1.2 Enumeration Type Documentation	92
7.1.2.1 anonymous enum	92
7.1.3 Function Documentation	92
7.1.3.1 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [1/34]	92
7.1.3.2 GINAC_BIND_UNARCHIVER() [1/50]	93
7.1.3.3 GINAC_DECLARE_UNARCHIVER() [1/52]	93
7.1.3.4 write_unsigned()	93
7.1.3.5 read_unsigned()	93
7.1.3.6 operator<<() [1/16]	93
7.1.3.7 operator<<() [2/16]	93
7.1.3.8 operator>>() [1/3]	94
7.1.3.9 operator>>() [2/3]	94
7.1.3.10 find_factory_fcn()	94
7.1.3.11 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [2/34]	94
7.1.3.12 is_a() [1/3]	95



7.1.3.13 <code>is_exactly_a()</code> [1/2]	95
7.1.3.14 <code>dynallocate()</code> [1/2]	96
7.1.3.15 <code>dynallocate()</code> [2/2]	97
7.1.3.16 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [3/34]	97
7.1.3.17 <code>print_func&lt; print_dflt &gt;()</code> [1/3]	97
7.1.3.18 <code>GINAC_BIND_UNARCHIVER()</code> [2/50]	98
7.1.3.19 <code>GINAC_BIND_UNARCHIVER()</code> [3/50]	98
7.1.3.20 <code>GINAC_BIND_UNARCHIVER()</code> [4/50]	98
7.1.3.21 <code>GINAC_BIND_UNARCHIVER()</code> [5/50]	98
7.1.3.22 <code>GINAC_BIND_UNARCHIVER()</code> [6/50]	98
7.1.3.23 <code>GINAC_BIND_UNARCHIVER()</code> [7/50]	98
7.1.3.24 <code>GINAC_BIND_UNARCHIVER()</code> [8/50]	98
7.1.3.25 <code>is_dirac_slash()</code>	98
7.1.3.26 <code>base_and_index()</code>	99
7.1.3.27 <code>dirac_ONE()</code>	99
7.1.3.28 <code>get_dim_uint()</code>	99
7.1.3.29 <code>clifford_unit()</code>	99
7.1.3.30 <code>dirac_gamma()</code>	100
7.1.3.31 <code>dirac_gamma5()</code>	100
7.1.3.32 <code>dirac_gammaL()</code>	101
7.1.3.33 <code>dirac_gammaR()</code>	101
7.1.3.34 <code>dirac_slash()</code>	101
7.1.3.35 <code>get_representation_label()</code> [1/2]	102
7.1.3.36 <code>trace_string()</code>	102
7.1.3.37 <code>dirac_trace()</code> [1/3]	102
7.1.3.38 <code>dirac_trace()</code> [2/3]	103
7.1.3.39 <code>dirac_trace()</code> [3/3]	103
7.1.3.40 <code>canonicalize_clifford()</code>	103
7.1.3.41 <code>clifford_star_bar()</code>	104
7.1.3.42 <code>clifford_prime()</code>	104
7.1.3.43 <code>remove_dirac_ONE()</code>	104
7.1.3.44 <code>clifford_max_label()</code>	104
7.1.3.45 <code>clifford_norm()</code>	105
7.1.3.46 <code>clifford_inverse()</code>	105
7.1.3.47 <code>lst_to_clifford()</code> [1/2]	105
7.1.3.48 <code>lst_to_clifford()</code> [2/2]	106
7.1.3.49 <code>get_clifford_comp()</code>	107
7.1.3.50 <code>clifford_to_lst()</code>	107
7.1.3.51 <code>clifford_moebius_map()</code> [1/2]	108
7.1.3.52 <code>clifford_moebius_map()</code> [2/2]	108
7.1.3.53 <code>GINAC_DECLARE_UNARCHIVER()</code> [2/52]	109
7.1.3.54 <code>GINAC_DECLARE_UNARCHIVER()</code> [3/52]	109

7.1.3.55 GINAC_DECLARE_UNARCHIVER() [4/52]	109
7.1.3.56 GINAC_DECLARE_UNARCHIVER() [5/52]	109
7.1.3.57 GINAC_DECLARE_UNARCHIVER() [6/52]	109
7.1.3.58 GINAC_DECLARE_UNARCHIVER() [7/52]	109
7.1.3.59 GINAC_DECLARE_UNARCHIVER() [8/52]	109
7.1.3.60 is_clifford_tinfo()	110
7.1.3.61 clifford_bar()	110
7.1.3.62 clifford_star()	110
7.1.3.63 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [4/34]	110
7.1.3.64 print_func< print_dflt >() [2/3]	110
7.1.3.65 GINAC_BIND_UNARCHIVER() [9/50]	111
7.1.3.66 GINAC_BIND_UNARCHIVER() [10/50]	111
7.1.3.67 GINAC_BIND_UNARCHIVER() [11/50]	111
7.1.3.68 GINAC_BIND_UNARCHIVER() [12/50]	111
7.1.3.69 GINAC_BIND_UNARCHIVER() [13/50]	111
7.1.3.70 permute_free_index_to_front()	111
7.1.3.71 color_ONE()	112
7.1.3.72 color_T()	112
7.1.3.73 color_f()	112
7.1.3.74 color_d()	113
7.1.3.75 color_h()	113
7.1.3.76 is_color_tinfo()	114
7.1.3.77 get_representation_label() [2/2]	114
7.1.3.78 color_trace() [1/3]	114
7.1.3.79 color_trace() [2/3]	114
7.1.3.80 color_trace() [3/3]	115
7.1.3.81 GINAC_DECLARE_UNARCHIVER() [9/52]	115
7.1.3.82 GINAC_DECLARE_UNARCHIVER() [10/52]	115
7.1.3.83 GINAC_DECLARE_UNARCHIVER() [11/52]	115
7.1.3.84 GINAC_DECLARE_UNARCHIVER() [12/52]	115
7.1.3.85 GINAC_DECLARE_UNARCHIVER() [13/52]	115
7.1.3.86 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [5/34]	116
7.1.3.87 GINAC_BIND_UNARCHIVER() [14/50]	116
7.1.3.88 GINAC_DECLARE_UNARCHIVER() [14/52]	116
7.1.3.89 crc32()	116
7.1.3.90 are_ex_trivially_equal()	116
7.1.3.91 operator<<() [3/16]	117
7.1.3.92 operator<<() [4/16]	117
7.1.3.93 operator<<() [5/16]	117
7.1.3.94 nops() [1/2]	117
7.1.3.95 expand() [1/2]	117
7.1.3.96 conjugate()	117

7.1.3.97 real_part()	118
7.1.3.98 imag_part()	118
7.1.3.99 has()	118
7.1.3.100 find()	118
7.1.3.101 is_polynomial()	118
7.1.3.102 degree()	119
7.1.3.103 ldegree()	119
7.1.3.104 coeff()	119
7.1.3.105 numer() $[1/2]$	119
7.1.3.106 denom() $[1/2]$	119
7.1.3.107 numer_denom()	120
7.1.3.108 normal()	120
7.1.3.109 to_rational()	120
7.1.3.110 to_polynomial()	120
7.1.3.111 collect()	120
7.1.3.112 eval()	120
7.1.3.113 evalf() $[1/2]$	121
7.1.3.114 evalm()	121
7.1.3.115 eval_integ()	121
7.1.3.116 diff()	121
7.1.3.117 series()	121
7.1.3.118 match()	122
7.1.3.119 simplify_indexed() $[1/3]$	122
7.1.3.120 simplify_indexed() $[2/3]$	122
7.1.3.121 symmetrize() $[1/4]$	122
7.1.3.122 symmetrize() $[2/4]$	122
7.1.3.123 antisymmetrize() $[1/4]$	123
7.1.3.124 antisymmetrize() $[2/4]$	123
7.1.3.125 symmetrize_cyclic() $[1/4]$	123
7.1.3.126 symmetrize_cyclic() $[2/4]$	123
7.1.3.127 op()	123
7.1.3.128 lhs()	123
7.1.3.129 rhs()	124
7.1.3.130 is_zero() $[1/2]$	124
7.1.3.131 swap() $[1/2]$	124
7.1.3.132 subs() $[1/3]$	124
7.1.3.133 subs() $[2/3]$	124
7.1.3.134 subs() $[3/3]$	125
7.1.3.135 is_a() $[2/3]$	125
7.1.3.136 is_exactly_a() $[2/2]$	125
7.1.3.137 ex_to()	125
7.1.3.138 compile_ex() $[1/3]$	127

7.1.3.139 compile_ex() [2/3]	127
7.1.3.140 compile_ex() [3/3]	128
7.1.3.141 link_ex() [1/3]	128
7.1.3.142 link_ex() [2/3]	128
7.1.3.143 link_ex() [3/3]	129
7.1.3.144 unlink_ex()	129
7.1.3.145 swap() [2/2]	129
7.1.3.146 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [6/34]	129
7.1.3.147 conjugateepvector()	130
7.1.3.148 factor()	130
7.1.3.149 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [7/34]	130
7.1.3.150 GINAC_DECLARE_UNARCHIVER() [15/52]	131
7.1.3.151 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [8/34]	131
7.1.3.152 GINAC_BIND_UNARCHIVER() [15/50]	131
7.1.3.153 GINAC_DECLARE_UNARCHIVER() [16/52]	131
7.1.3.154 GINAC_BIND_UNARCHIVER() [16/50]	131
7.1.3.155 GINAC_DECLARE_UNARCHIVER() [17/52]	131
7.1.3.156 is_the_function()	131
7.1.3.157 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [9/34]	132
7.1.3.158 GINAC_BIND_UNARCHIVER() [17/50]	132
7.1.3.159 GINAC_DECLARE_UNARCHIVER() [18/52]	132
7.1.3.160 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [10/34]	132
7.1.3.161 print_func< print_context >()	132
7.1.3.162 GINAC_BIND_UNARCHIVER() [18/50]	132
7.1.3.163 GINAC_BIND_UNARCHIVER() [19/50]	132
7.1.3.164 GINAC_BIND_UNARCHIVER() [20/50]	133
7.1.3.165 is_dummy_pair() [1/2]	133
7.1.3.166 is_dummy_pair() [2/2]	133
7.1.3.167 find_free_and_dummy() [1/2]	133
7.1.3.168 minimal_dim()	134
7.1.3.169 GINAC_DECLARE_UNARCHIVER() [19/52]	134
7.1.3.170 GINAC_DECLARE_UNARCHIVER() [20/52]	134
7.1.3.171 GINAC_DECLARE_UNARCHIVER() [21/52]	134
7.1.3.172 find_free_and_dummy() [2/2]	134
7.1.3.173 find_dummy_indices()	135
7.1.3.174 count_dummy_indices()	136
7.1.3.175 count_free_indices()	136
7.1.3.176 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [11/34]	136
7.1.3.177 GINAC_BIND_UNARCHIVER() [21/50]	136
7.1.3.178 indices_consistent()	136
7.1.3.179 number_of_type()	137
7.1.3.180 rename_dummy_indices()	137

7.1.3.181 find_variant_indices()	137
7.1.3.182 reposition_dummy_indices()	137
7.1.3.183 product_to_exvector()	138
7.1.3.184 idx_symmetrization()	138
7.1.3.185 simplify_indexed() [3/3]	138
7.1.3.186 simplify_indexed_product()	138
7.1.3.187 hasindex()	139
7.1.3.188 get_all_dummy_indices_safely()	139
7.1.3.189 get_all_dummy_indices()	139
7.1.3.190 rename_dummy_indices_uniquely() [1/4]	139
7.1.3.191 rename_dummy_indices_uniquely() [2/4]	140
7.1.3.192 rename_dummy_indices_uniquely() [3/4]	140
7.1.3.193 rename_dummy_indices_uniquely() [4/4]	140
7.1.3.194 expand_dummy_sum()	140
7.1.3.195 GINAC_DECLARE_UNARCHIVER() [22/52]	141
7.1.3.196 conjugate_evalf()	141
7.1.3.197 conjugate_eval()	141
7.1.3.198 conjugate_print_latex()	141
7.1.3.199 conjugate_conjugate()	141
7.1.3.200 conjugate_expl_derivative()	142
7.1.3.201 conjugate_real_part()	142
7.1.3.202 conjugate_imag_part()	142
7.1.3.203 func_arg_info()	142
7.1.3.204 conjugate_info()	142
7.1.3.205 REGISTER_FUNCTION() [1/36]	143
7.1.3.206 real_part_evalf()	143
7.1.3.207 real_part_eval()	143
7.1.3.208 real_part_print_latex()	143
7.1.3.209 real_part_conjugate()	143
7.1.3.210 real_part_real_part()	144
7.1.3.211 real_part_imag_part()	144
7.1.3.212 real_part_expl_derivative()	144
7.1.3.213 REGISTER_FUNCTION() [2/36]	144
7.1.3.214 imag_part_evalf()	144
7.1.3.215 imag_part_eval()	145
7.1.3.216 imag_part_print_latex()	145
7.1.3.217 imag_part_conjugate()	145
7.1.3.218 imag_part_real_part()	145
7.1.3.219 imag_part_imag_part()	145
7.1.3.220 imag_part_expl_derivative()	145
7.1.3.221 REGISTER_FUNCTION() [3/36]	146
7.1.3.222 abs_evalf()	146

7.1.3.223 <code>abs_eval()</code>	146
7.1.3.224 <code>abs_expand()</code>	146
7.1.3.225 <code>abs_expl_derivative()</code>	146
7.1.3.226 <code>abs_print_latex()</code>	147
7.1.3.227 <code>abs_print_csrc_float()</code>	147
7.1.3.228 <code>abs_conjugate()</code>	147
7.1.3.229 <code>abs_real_part()</code>	147
7.1.3.230 <code>abs_imag_part()</code>	147
7.1.3.231 <code>abs_power()</code>	148
7.1.3.232 <code>abs_info()</code>	148
7.1.3.233 <code>REGISTER_FUNCTION()</code> [4/36]	148
7.1.3.234 <code>step_evalf()</code>	148
7.1.3.235 <code>step_eval()</code>	148
7.1.3.236 <code>step_series()</code>	149
7.1.3.237 <code>step_conjugate()</code>	149
7.1.3.238 <code>step_real_part()</code>	149
7.1.3.239 <code>step_imag_part()</code>	149
7.1.3.240 <code>REGISTER_FUNCTION()</code> [5/36]	149
7.1.3.241 <code>csgn_evalf()</code>	150
7.1.3.242 <code>csgn_eval()</code>	150
7.1.3.243 <code>csgn_series()</code>	150
7.1.3.244 <code>csgn_conjugate()</code>	150
7.1.3.245 <code>csgn_real_part()</code>	150
7.1.3.246 <code>csgn_imag_part()</code>	151
7.1.3.247 <code>csgn_power()</code>	151
7.1.3.248 <code>REGISTER_FUNCTION()</code> [6/36]	151
7.1.3.249 <code>eta_evalf()</code>	151
7.1.3.250 <code>eta_eval()</code>	151
7.1.3.251 <code>eta_series()</code>	152
7.1.3.252 <code>eta_conjugate()</code>	152
7.1.3.253 <code>eta_real_part()</code>	152
7.1.3.254 <code>eta_imag_part()</code>	152
7.1.3.255 <code>REGISTER_FUNCTION()</code> [7/36]	152
7.1.3.256 <code>Li2_evalf()</code>	153
7.1.3.257 <code>Li2_eval()</code>	153
7.1.3.258 <code>Li2_deriv()</code>	153
7.1.3.259 <code>Li2_series()</code> [1/2]	153
7.1.3.260 <code>Li2_conjugate()</code>	153
7.1.3.261 <code>REGISTER_FUNCTION()</code> [8/36]	154
7.1.3.262 <code>Li3_eval()</code>	154
7.1.3.263 <code>REGISTER_FUNCTION()</code> [9/36]	154
7.1.3.264 <code>zetaderiv_eval()</code>	154

7.1.3.265 zetaderiv_deriv()	154
7.1.3.266 REGISTER_FUNCTION() [10/36]	155
7.1.3.267 factorial_evalf()	155
7.1.3.268 factorial_eval()	155
7.1.3.269 factorial_print_dfit_latex()	155
7.1.3.270 factorial_conjugate()	155
7.1.3.271 factorial_real_part()	156
7.1.3.272 factorial_imag_part()	156
7.1.3.273 REGISTER_FUNCTION() [11/36]	156
7.1.3.274 binomial_evalf()	156
7.1.3.275 binomial_sym()	156
7.1.3.276 binomial_eval()	157
7.1.3.277 binomial_conjugate()	157
7.1.3.278 binomial_real_part()	157
7.1.3.279 binomial_imag_part()	157
7.1.3.280 REGISTER_FUNCTION() [12/36]	157
7.1.3.281 Order_eval()	158
7.1.3.282 Order_series()	158
7.1.3.283 Order_conjugate()	158
7.1.3.284 Order_real_part()	158
7.1.3.285 Order_imag_part()	158
7.1.3.286 Order_power()	158
7.1.3.287 Order_expl_derivative()	159
7.1.3.288 REGISTER_FUNCTION() [13/36]	159
7.1.3.289 lsolve()	159
7.1.3.290 fsolve()	159
7.1.3.291 zeta() [1/3]	160
7.1.3.292 zeta() [2/3]	160
7.1.3.293 is_the_function< zeta_SERIAL >()	160
7.1.3.294 G() [1/2]	160
7.1.3.295 G() [2/2]	161
7.1.3.296 is_the_function< G_SERIAL >()	161
7.1.3.297 psi() [1/4]	161
7.1.3.298 psi() [2/4]	161
7.1.3.299 is_the_function< psi_SERIAL >()	161
7.1.3.300 iterated_integral() [1/2]	162
7.1.3.301 iterated_integral() [2/2]	162
7.1.3.302 is_the_function< iterated_integral_SERIAL >()	162
7.1.3.303 is_order_function()	162
7.1.3.304 convert_H_to_Li()	162
7.1.3.305 EllipticK_evalf()	163
7.1.3.306 EllipticK_eval()	163

7.1.3.307 EllipticK_deriv()	163
7.1.3.308 EllipticK_series()	163
7.1.3.309 EllipticK_print_latex()	163
7.1.3.310 REGISTER_FUNCTION() [14/36]	164
7.1.3.311 EllipticE_evalf()	164
7.1.3.312 EllipticE_eval()	164
7.1.3.313 EllipticE_deriv()	164
7.1.3.314 EllipticE_series()	164
7.1.3.315 EllipticE_print_latex()	165
7.1.3.316 REGISTER_FUNCTION() [15/36]	165
7.1.3.317 iterated_integral_evalf_impl()	165
7.1.3.318 iterated_integral2_evalf()	165
7.1.3.319 iterated_integral3_evalf()	165
7.1.3.320 iterated_integral2_eval()	166
7.1.3.321 iterated_integral3_eval()	166
7.1.3.322 lgamma_evalf()	166
7.1.3.323 lgamma_eval()	166
7.1.3.324 lgamma_deriv()	167
7.1.3.325 lgamma_series()	167
7.1.3.326 lgamma_conjugate()	167
7.1.3.327 REGISTER_FUNCTION() [16/36]	167
7.1.3.328 tgamma_evalf()	167
7.1.3.329 tgamma_eval()	168
7.1.3.330 tgamma_deriv()	168
7.1.3.331 tgamma_series()	168
7.1.3.332 tgamma_conjugate()	168
7.1.3.333 REGISTER_FUNCTION() [17/36]	169
7.1.3.334 beta_evalf()	169
7.1.3.335 beta_eval()	169
7.1.3.336 beta_deriv()	169
7.1.3.337 beta_series()	169
7.1.3.338 REGISTER_FUNCTION() [18/36]	170
7.1.3.339 psi1_evalf()	170
7.1.3.340 psi1_eval()	170
7.1.3.341 psi1_deriv()	170
7.1.3.342 psi1_series()	170
7.1.3.343 psi2_evalf()	171
7.1.3.344 psi2_eval()	171
7.1.3.345 psi2_deriv()	171
7.1.3.346 psi2_series()	171
7.1.3.347 G2_evalf()	171
7.1.3.348 G2_eval()	172



7.1.3.349 G3_evalf()	172
7.1.3.350 G3_eval()	172
7.1.3.351 Li_evalf()	172
7.1.3.352 Li_eval()	172
7.1.3.353 Li_series()	173
7.1.3.354 Li_deriv()	173
7.1.3.355 Li_print_latex()	173
7.1.3.356 REGISTER_FUNCTION() [19/36]	173
7.1.3.357 S_evalf()	173
7.1.3.358 S_eval()	174
7.1.3.359 S_series()	174
7.1.3.360 S_deriv()	174
7.1.3.361 S_print_latex()	174
7.1.3.362 REGISTER_FUNCTION() [20/36]	175
7.1.3.363 H_evalf()	175
7.1.3.364 H_eval()	175
7.1.3.365 H_series()	175
7.1.3.366 H_deriv()	175
7.1.3.367 H_print_latex()	176
7.1.3.368 REGISTER_FUNCTION() [21/36]	176
7.1.3.369 zeta1_evalf()	176
7.1.3.370 zeta1_eval()	176
7.1.3.371 zeta1_deriv()	176
7.1.3.372 zeta1_print_latex()	177
7.1.3.373 zeta2_evalf()	177
7.1.3.374 zeta2_eval()	177
7.1.3.375 zeta2_deriv()	177
7.1.3.376 zeta2_print_latex()	177
7.1.3.377 exp_evalf()	177
7.1.3.378 exp_eval()	178
7.1.3.379 exp_expand()	178
7.1.3.380 exp_deriv()	178
7.1.3.381 exp_real_part()	178
7.1.3.382 exp_imag_part()	178
7.1.3.383 exp_conjugate()	179
7.1.3.384 exp_power()	179
7.1.3.385 exp_info()	179
7.1.3.386 REGISTER_FUNCTION() [22/36]	179
7.1.3.387 log_evalf()	179
7.1.3.388 log_eval()	180
7.1.3.389 log_deriv()	180
7.1.3.390 log_series()	180

7.1.3.391 log_real_part()	180
7.1.3.392 log_imag_part()	180
7.1.3.393 log_expand()	181
7.1.3.394 log_conjugate()	181
7.1.3.395 log_info()	181
7.1.3.396 REGISTER_FUNCTION() [23/36]	181
7.1.3.397 sin_evalf()	181
7.1.3.398 sin_eval()	182
7.1.3.399 sin_deriv()	182
7.1.3.400 sin_real_part()	182
7.1.3.401 sin_imag_part()	182
7.1.3.402 sin_conjugate()	182
7.1.3.403 trig_info()	183
7.1.3.404 REGISTER_FUNCTION() [24/36]	183
7.1.3.405 cos_evalf()	183
7.1.3.406 cos_eval()	183
7.1.3.407 cos_deriv()	183
7.1.3.408 cos_real_part()	184
7.1.3.409 cos_imag_part()	184
7.1.3.410 cos_conjugate()	184
7.1.3.411 REGISTER_FUNCTION() [25/36]	184
7.1.3.412 tan_evalf()	184
7.1.3.413 tan_eval()	185
7.1.3.414 tan_deriv()	185
7.1.3.415 tan_real_part()	185
7.1.3.416 tan_imag_part()	185
7.1.3.417 tan_series()	185
7.1.3.418 tan_conjugate()	186
7.1.3.419 REGISTER_FUNCTION() [26/36]	186
7.1.3.420 asin_evalf()	186
7.1.3.421 asin_eval()	186
7.1.3.422 asin_deriv()	186
7.1.3.423 asin_conjugate()	187
7.1.3.424 asin_info()	187
7.1.3.425 REGISTER_FUNCTION() [27/36]	187
7.1.3.426 acos_evalf()	187
7.1.3.427 acos_eval()	187
7.1.3.428 acos_deriv()	188
7.1.3.429 acos_conjugate()	188
7.1.3.430 REGISTER_FUNCTION() [28/36]	188
7.1.3.431 atan_evalf()	188
7.1.3.432 atan_eval()	188

7.1.3.433 atan_deriv()	189
7.1.3.434 atan_series()	189
7.1.3.435 atan_conjugate()	189
7.1.3.436 atan_info()	189
7.1.3.437 REGISTER_FUNCTION() [29/36]	189
7.1.3.438 atan2_evalf()	190
7.1.3.439 atan2_eval()	190
7.1.3.440 atan2_deriv()	190
7.1.3.441 atan2_info()	190
7.1.3.442 REGISTER_FUNCTION() [30/36]	190
7.1.3.443 sinh_evalf()	191
7.1.3.444 sinh_eval()	191
7.1.3.445 sinh_deriv()	191
7.1.3.446 sinh_real_part()	191
7.1.3.447 sinh_imag_part()	191
7.1.3.448 sinh_conjugate()	192
7.1.3.449 REGISTER_FUNCTION() [31/36]	192
7.1.3.450 cosh_evalf()	192
7.1.3.451 cosh_eval()	192
7.1.3.452 cosh_deriv()	192
7.1.3.453 cosh_real_part()	193
7.1.3.454 cosh_imag_part()	193
7.1.3.455 cosh_conjugate()	193
7.1.3.456 REGISTER_FUNCTION() [32/36]	193
7.1.3.457 tanh_evalf()	193
7.1.3.458 tanh_eval()	194
7.1.3.459 tanh_deriv()	194
7.1.3.460 tanh_series()	194
7.1.3.461 tanh_real_part()	194
7.1.3.462 tanh_imag_part()	194
7.1.3.463 tanh_conjugate()	195
7.1.3.464 REGISTER_FUNCTION() [33/36]	195
7.1.3.465 asinh_evalf()	195
7.1.3.466 asinh_eval()	195
7.1.3.467 asinh_deriv()	195
7.1.3.468 asinh_conjugate()	196
7.1.3.469 REGISTER_FUNCTION() [34/36]	196
7.1.3.470 acosh_evalf()	196
7.1.3.471 acosh_eval()	196
7.1.3.472 acosh_deriv()	196
7.1.3.473 acosh_conjugate()	197
7.1.3.474 REGISTER_FUNCTION() [35/36]	197

7.1.3.475 <code>atanh_evalf()</code> . . . . .	197
7.1.3.476 <code>atanh_eval()</code> . . . . .	197
7.1.3.477 <code>atanh_deriv()</code> . . . . .	197
7.1.3.478 <code>atanh_series()</code> . . . . .	198
7.1.3.479 <code>atanh_conjugate()</code> . . . . .	198
7.1.3.480 <code>REGISTER_FUNCTION()</code> [36/36] . . . . .	198
7.1.3.481 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [12/34] . . . . .	198
7.1.3.482 <code>subvalue()</code> . . . . .	198
7.1.3.483 <code>adaptivesimpson()</code> . . . . .	199
7.1.3.484 <code>GINAC_BIND_UNARCHIVER()</code> [22/50] . . . . .	199
7.1.3.485 <code>GINAC_DECLARE_UNARCHIVER()</code> [23/52] . . . . .	199
7.1.3.486 <code>ifactor()</code> . . . . .	199
7.1.3.487 <code>is_discriminant_of_quadratic_number_field()</code> . . . . .	200
7.1.3.488 <code>kronecker_symbol()</code> . . . . .	200
7.1.3.489 <code>primitive_dirichlet_character()</code> . . . . .	200
7.1.3.490 <code>dirichlet_character()</code> . . . . .	201
7.1.3.491 <code>generalised_Bernoulli_number()</code> . . . . .	201
7.1.3.492 <code>Bernoulli_polynomial()</code> . . . . .	201
7.1.3.493 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [13/34] . . . . .	201
7.1.3.494 <code>GINAC_BIND_UNARCHIVER()</code> [23/50] . . . . .	202
7.1.3.495 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [14/34] . . . . .	202
7.1.3.496 <code>GINAC_BIND_UNARCHIVER()</code> [24/50] . . . . .	202
7.1.3.497 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [15/34] . . . . .	202
7.1.3.498 <code>GINAC_BIND_UNARCHIVER()</code> [25/50] . . . . .	202
7.1.3.499 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [16/34] . . . . .	202
7.1.3.500 <code>GINAC_BIND_UNARCHIVER()</code> [26/50] . . . . .	202
7.1.3.501 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [17/34] . . . . .	203
7.1.3.502 <code>GINAC_BIND_UNARCHIVER()</code> [27/50] . . . . .	203
7.1.3.503 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [18/34] . . . . .	203
7.1.3.504 <code>GINAC_BIND_UNARCHIVER()</code> [28/50] . . . . .	203
7.1.3.505 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [19/34] . . . . .	203
7.1.3.506 <code>GINAC_BIND_UNARCHIVER()</code> [29/50] . . . . .	203
7.1.3.507 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [20/34] . . . . .	203
7.1.3.508 <code>GINAC_BIND_UNARCHIVER()</code> [30/50] . . . . .	204
7.1.3.509 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [21/34] . . . . .	204
7.1.3.510 <code>GINAC_BIND_UNARCHIVER()</code> [31/50] . . . . .	204
7.1.3.511 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [22/34] . . . . .	204
7.1.3.512 <code>GINAC_BIND_UNARCHIVER()</code> [32/50] . . . . .	204
7.1.3.513 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [23/34] . . . . .	204
7.1.3.514 <code>GINAC_BIND_UNARCHIVER()</code> [33/50] . . . . .	204
7.1.3.515 <code>GINAC_DECLARE_UNARCHIVER()</code> [24/52] . . . . .	205
7.1.3.516 <code>GINAC_DECLARE_UNARCHIVER()</code> [25/52] . . . . .	205

7.1.3.517 GINAC_DECLARE_UNARCHIVER() [26/52]	205
7.1.3.518 GINAC_DECLARE_UNARCHIVER() [27/52]	205
7.1.3.519 GINAC_DECLARE_UNARCHIVER() [28/52]	205
7.1.3.520 GINAC_DECLARE_UNARCHIVER() [29/52]	205
7.1.3.521 GINAC_DECLARE_UNARCHIVER() [30/52]	205
7.1.3.522 GINAC_DECLARE_UNARCHIVER() [31/52]	205
7.1.3.523 GINAC_DECLARE_UNARCHIVER() [32/52]	205
7.1.3.524 GINAC_DECLARE_UNARCHIVER() [33/52]	206
7.1.3.525 GINAC_DECLARE_UNARCHIVER() [34/52]	206
7.1.3.526 GINAC_DECLARE_UNARCHIVER() [35/52]	206
7.1.3.527 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [24/34]	206
7.1.3.528 GINAC_BIND_UNARCHIVER() [34/50]	206
7.1.3.529 lst_to_matrix()	206
7.1.3.530 diag_matrix() [1/2]	207
7.1.3.531 diag_matrix() [2/2]	207
7.1.3.532 unit_matrix() [1/2]	207
7.1.3.533 symbolic_matrix() [1/2]	207
7.1.3.534 reduced_matrix()	208
7.1.3.535 sub_matrix()	208
7.1.3.536 GINAC_DECLARE_UNARCHIVER() [36/52]	208
7.1.3.537 nops() [2/2]	208
7.1.3.538 expand() [2/2]	208
7.1.3.539 evalf() [2/2]	209
7.1.3.540 rows()	209
7.1.3.541 cols()	209
7.1.3.542 transpose()	209
7.1.3.543 determinant()	209
7.1.3.544 trace()	209
7.1.3.545 charpoly()	210
7.1.3.546 inverse() [1/3]	210
7.1.3.547 inverse() [2/3]	210
7.1.3.548 rank() [1/2]	210
7.1.3.549 rank() [2/2]	210
7.1.3.550 unit_matrix() [2/2]	210
7.1.3.551 symbolic_matrix() [2/2]	211
7.1.3.552 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [25/34]	211
7.1.3.553 tryfactsubs()	211
7.1.3.554 algebraic_match_mul_with_mul()	211
7.1.3.555 GINAC_BIND_UNARCHIVER() [35/50]	212
7.1.3.556 GINAC_DECLARE_UNARCHIVER() [37/52]	212
7.1.3.557 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [26/34]	212
7.1.3.558 reeval_ncmul()	212

7.1.3.559 hold_ncmul()	212
7.1.3.560 GINAC_BIND_UNARCHIVER() [36/50]	212
7.1.3.561 GINAC_DECLARE_UNARCHIVER() [38/52]	212
7.1.3.562 get_first_symbol()	213
7.1.3.563 add_symbol()	214
7.1.3.564 collect_symbols()	214
7.1.3.565 get_symbol_stats()	214
7.1.3.566 lcmcoeff()	215
7.1.3.567 lcm_of_coefficients_denominators()	215
7.1.3.568 multiply_lcm()	215
7.1.3.569 quo()	216
7.1.3.570 rem()	216
7.1.3.571 decomp_rational()	217
7.1.3.572 prem()	217
7.1.3.573 sprem()	217
7.1.3.574 divide()	218
7.1.3.575 divide_in_z()	218
7.1.3.576 sr_gcd()	219
7.1.3.577 interpolate()	220
7.1.3.578 heur_gcd_z()	220
7.1.3.579 heur_gcd()	221
7.1.3.580 gcd_pf_pow()	221
7.1.3.581 gcd_pf_mul()	222
7.1.3.582 gcd() [1/2]	222
7.1.3.583 gcd_pf_pow_pow()	223
7.1.3.584 lcm() [1/2]	223
7.1.3.585 sqrfree_yun()	223
7.1.3.586 sqrfree()	224
7.1.3.587 sqrfree_parfrac()	224
7.1.3.588 replace_with_symbol() [1/2]	225
7.1.3.589 replace_with_symbol() [2/2]	226
7.1.3.590 frac_cancel()	226
7.1.3.591 find_common_factor()	226
7.1.3.592 collect_common_factors()	227
7.1.3.593 resultant()	227
7.1.3.594 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [27/34]	227
7.1.3.595 make_real_float()	227
7.1.3.596 read_real_float()	228
7.1.3.597 GINAC_BIND_UNARCHIVER() [37/50]	228
7.1.3.598 write_real_float()	228
7.1.3.599 print_real_number()	228
7.1.3.600 print_integer_csrc()	229

7.1.3.601 <code>print_real_csrc()</code>	229
7.1.3.602 <code>coerce()</code>	229
7.1.3.603 <code>coerce&lt; int, cln::cl_I &gt;()</code>	229
7.1.3.604 <code>coerce&lt; unsigned int, cln::cl_I &gt;()</code>	230
7.1.3.605 <code>print_real_cl_N()</code>	230
7.1.3.606 <code>exp()</code>	230
7.1.3.607 <code>log()</code>	231
7.1.3.608 <code>sin()</code>	231
7.1.3.609 <code>cos()</code>	231
7.1.3.610 <code>tan()</code>	232
7.1.3.611 <code>asin()</code>	232
7.1.3.612 <code>acos()</code>	232
7.1.3.613 <code>atan()</code> $[1/2]$	233
7.1.3.614 <code>atan()</code> $[2/2]$	233
7.1.3.615 <code>sinh()</code>	234
7.1.3.616 <code>cosh()</code>	234
7.1.3.617 <code>tanh()</code>	234
7.1.3.618 <code>asinh()</code>	235
7.1.3.619 <code>acosh()</code>	235
7.1.3.620 <code>atanh()</code>	235
7.1.3.621 <code>Li2_series()</code> $[2/2]$	236
7.1.3.622 <code>Li2_projection()</code>	236
7.1.3.623 <code>Li2_()</code>	236
7.1.3.624 <code>Li2()</code>	236
7.1.3.625 <code>zeta()</code> $[3/3]$	237
7.1.3.626 <code>guess_precision()</code>	237
7.1.3.627 <code>lgamma()</code> $[1/2]$	237
7.1.3.628 <code>lgamma()</code> $[2/2]$	237
7.1.3.629 <code>tgamma()</code> $[1/2]$	237
7.1.3.630 <code>tgamma()</code> $[2/2]$	238
7.1.3.631 <code>psi()</code> $[3/4]$	238
7.1.3.632 <code>psi()</code> $[4/4]$	238
7.1.3.633 <code>factorial()</code>	238
7.1.3.634 <code>doublefactorial()</code>	239
7.1.3.635 <code>binomial()</code>	239
7.1.3.636 <code>bernoulli()</code>	240
7.1.3.637 <code>fibonacci()</code>	240
7.1.3.638 <code>abs()</code>	241
7.1.3.639 <code>mod()</code>	241
7.1.3.640 <code>smod()</code>	241
7.1.3.641 <code>irem()</code> $[1/2]$	242
7.1.3.642 <code>irem()</code> $[2/2]$	242

7.1.3.643 <code>iquo()</code> [1/2]	243
7.1.3.644 <code>iquo()</code> [2/2]	243
7.1.3.645 <code>gcd()</code> [2/2]	243
7.1.3.646 <code>lcm()</code> [2/2]	244
7.1.3.647 <code>sqrt()</code> [1/2]	244
7.1.3.648 <code>isqrt()</code>	244
7.1.3.649 <code>PiEvalf()</code>	245
7.1.3.650 <code>EulerEvalf()</code>	245
7.1.3.651 <code>CatalanEvalf()</code>	245
7.1.3.652 <code>operator&lt;&lt;()</code> [6/16]	245
7.1.3.653 <code>GINAC_DECLARE_UNARCHIVER()</code> [39/52]	245
7.1.3.654 <code>pow()</code> [1/3]	245
7.1.3.655 <code>inverse()</code> [3/3]	246
7.1.3.656 <code>step()</code>	246
7.1.3.657 <code>csgn()</code>	246
7.1.3.658 <code>is_zero()</code> [2/2]	246
7.1.3.659 <code>is_positive()</code>	246
7.1.3.660 <code>is_negative()</code>	246
7.1.3.661 <code>is_integer()</code>	247
7.1.3.662 <code>is_pos_integer()</code>	247
7.1.3.663 <code>is_nonneg_integer()</code>	247
7.1.3.664 <code>is_even()</code>	247
7.1.3.665 <code>is_odd()</code>	247
7.1.3.666 <code>is_prime()</code>	247
7.1.3.667 <code>is_rational()</code>	248
7.1.3.668 <code>is_real()</code>	248
7.1.3.669 <code>is_cinteger()</code>	248
7.1.3.670 <code>is_crational()</code>	248
7.1.3.671 <code>to_int()</code>	248
7.1.3.672 <code>to_long()</code>	248
7.1.3.673 <code>to_double()</code>	249
7.1.3.674 <code>real()</code>	249
7.1.3.675 <code>imag()</code>	249
7.1.3.676 <code>numer()</code> [2/2]	249
7.1.3.677 <code>denom()</code> [2/2]	249
7.1.3.678 <code>exadd()</code>	249
7.1.3.679 <code>exmul()</code>	250
7.1.3.680 <code>exminus()</code>	250
7.1.3.681 <code>operator+()</code> [1/4]	250
7.1.3.682 <code>operator-()</code> [1/4]	250
7.1.3.683 <code>operator*()</code> [1/2]	250
7.1.3.684 <code>operator/()</code> [1/2]	251



7.1.3.685 operator+() [2/4]	251
7.1.3.686 operator-() [2/4]	251
7.1.3.687 operator*() [2/2]	251
7.1.3.688 operator/() [2/2]	251
7.1.3.689 operator+=() [1/2]	251
7.1.3.690 operator-=() [1/2]	252
7.1.3.691 operator*=() [1/2]	252
7.1.3.692 operator/=() [1/2]	252
7.1.3.693 operator+=() [2/2]	252
7.1.3.694 operator-=() [2/2]	252
7.1.3.695 operator*=() [2/2]	252
7.1.3.696 operator/=() [2/2]	253
7.1.3.697 operator+() [3/4]	253
7.1.3.698 operator-() [3/4]	253
7.1.3.699 operator+() [4/4]	253
7.1.3.700 operator-() [4/4]	253
7.1.3.701 operator++() [1/4]	253
7.1.3.702 operator--() [1/4]	254
7.1.3.703 operator++() [2/4]	254
7.1.3.704 operator--() [2/4]	254
7.1.3.705 operator++() [3/4]	254
7.1.3.706 operator--() [3/4]	255
7.1.3.707 operator++() [4/4]	255
7.1.3.708 operator--() [4/4]	255
7.1.3.709 operator==(())	255
7.1.3.710 operator!=(())	255
7.1.3.711 operator<()	256
7.1.3.712 operator<=()	256
7.1.3.713 operator>()	256
7.1.3.714 operator>=()	256
7.1.3.715 my_ios_index()	256
7.1.3.716 my_ios_callback()	256
7.1.3.717 get_print_context()	257
7.1.3.718 set_print_context()	257
7.1.3.719 get_print_options()	257
7.1.3.720 set_print_options()	257
7.1.3.721 operator<<() [7/16]	257
7.1.3.722 operator>>() [3/3]	258
7.1.3.723 dflit()	258
7.1.3.724 latex()	258
7.1.3.725 python()	258
7.1.3.726 python_repr()	258

7.1.3.727 tree()	258
7.1.3.728 csrc()	258
7.1.3.729 csrc_float()	259
7.1.3.730 csrc_double()	259
7.1.3.731 csrc_cl_N()	259
7.1.3.732 index_dimensions()	259
7.1.3.733 no_index_dimensions()	259
7.1.3.734 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [28/34]	259
7.1.3.735 print_sym_pow()	260
7.1.3.736 GINAC_BIND_UNARCHIVER() [38/50]	260
7.1.3.737 GINAC_DECLARE_UNARCHIVER() [40/52]	260
7.1.3.738 pow() [2/3]	260
7.1.3.739 pow() [3/3]	260
7.1.3.740 sqrt() [2/2]	261
7.1.3.741 is_a() [3/3]	261
7.1.3.742 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [29/34]	261
7.1.3.743 GINAC_BIND_UNARCHIVER() [39/50]	261
7.1.3.744 GINAC_DECLARE_UNARCHIVER() [41/52]	261
7.1.3.745 series_to_poly()	262
7.1.3.746 is_terminating()	262
7.1.3.747 make_return_type_t()	262
7.1.3.748 set_print_func() [1/2]	262
7.1.3.749 set_print_func() [2/2]	263
7.1.3.750 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [30/34]	263
7.1.3.751 GINAC_BIND_UNARCHIVER() [40/50]	263
7.1.3.752 print_operator()	263
7.1.3.753 GINAC_DECLARE_UNARCHIVER() [42/52]	263
7.1.3.754 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [31/34]	263
7.1.3.755 get_default_TeX_name()	264
7.1.3.756 GINAC_BIND_UNARCHIVER() [41/50]	264
7.1.3.757 GINAC_BIND_UNARCHIVER() [42/50]	264
7.1.3.758 GINAC_BIND_UNARCHIVER() [43/50]	264
7.1.3.759 GINAC_DECLARE_UNARCHIVER() [43/52]	264
7.1.3.760 GINAC_DECLARE_UNARCHIVER() [44/52]	264
7.1.3.761 GINAC_DECLARE_UNARCHIVER() [45/52]	264
7.1.3.762 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [32/34]	265
7.1.3.763 GINAC_BIND_UNARCHIVER() [44/50]	265
7.1.3.764 index0()	265
7.1.3.765 index1()	265
7.1.3.766 index2()	265
7.1.3.767 index3()	265
7.1.3.768 not_symmetric()	266

7.1.3.769 symmetric2()	266
7.1.3.770 symmetric3()	266
7.1.3.771 symmetric4()	266
7.1.3.772 antisymmetric2()	266
7.1.3.773 antisymmetric3()	266
7.1.3.774 antisymmetric4()	267
7.1.3.775 canonicalize()	267
7.1.3.776 symm()	267
7.1.3.777 symmetrize() [3/4]	267
7.1.3.778 antisymmetrize() [3/4]	268
7.1.3.779 symmetrize_cyclic() [3/4]	268
7.1.3.780 GINAC_DECLARE_UNARCHIVER() [46/52]	268
7.1.3.781 sy_none() [1/4]	268
7.1.3.782 sy_none() [2/4]	268
7.1.3.783 sy_none() [3/4]	268
7.1.3.784 sy_none() [4/4]	269
7.1.3.785 sy_symm() [1/4]	269
7.1.3.786 sy_symm() [2/4]	269
7.1.3.787 sy_symm() [3/4]	269
7.1.3.788 sy_symm() [4/4]	269
7.1.3.789 sy_anti() [1/4]	269
7.1.3.790 sy_anti() [2/4]	270
7.1.3.791 sy_anti() [3/4]	270
7.1.3.792 sy_anti() [4/4]	270
7.1.3.793 sy_cycl() [1/4]	270
7.1.3.794 sy_cycl() [2/4]	270
7.1.3.795 sy_cycl() [3/4]	270
7.1.3.796 sy_cycl() [4/4]	271
7.1.3.797 symmetrize() [4/4]	271
7.1.3.798 antisymmetrize() [4/4]	271
7.1.3.799 symmetrize_cyclic() [4/4]	271
7.1.3.800 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [33/34]	271
7.1.3.801 print_func< print_dflt >() [3/3]	272
7.1.3.802 GINAC_BIND_UNARCHIVER() [45/50]	272
7.1.3.803 GINAC_BIND_UNARCHIVER() [46/50]	272
7.1.3.804 GINAC_BIND_UNARCHIVER() [47/50]	272
7.1.3.805 GINAC_BIND_UNARCHIVER() [48/50]	272
7.1.3.806 GINAC_BIND_UNARCHIVER() [49/50]	272
7.1.3.807 delta_tensor()	272
7.1.3.808 metric_tensor()	273
7.1.3.809 lorentz_g()	273
7.1.3.810 spinor_metric()	274

7.1.3.811 epsilon_tensor() [1/2]	274
7.1.3.812 epsilon_tensor() [2/2]	275
7.1.3.813 lorentz_eps()	275
7.1.3.814 GINAC_DECLARE_UNARCHIVER() [47/52]	276
7.1.3.815 GINAC_DECLARE_UNARCHIVER() [48/52]	276
7.1.3.816 GINAC_DECLARE_UNARCHIVER() [49/52]	276
7.1.3.817 GINAC_DECLARE_UNARCHIVER() [50/52]	276
7.1.3.818 GINAC_DECLARE_UNARCHIVER() [51/52]	276
7.1.3.819 log2()	276
7.1.3.820 multinomial_coefficient()	277
7.1.3.821 rotate_left()	277
7.1.3.822 compare_pointers()	277
7.1.3.823 golden_ratio_hash()	277
7.1.3.824 permutation_sign() [1/2]	278
7.1.3.825 permutation_sign() [2/2]	278
7.1.3.826 shaker_sort()	278
7.1.3.827 cyclic_permutation()	278
7.1.3.828 format_index_value() [1/2]	279
7.1.3.829 format_index_value() [2/2]	279
7.1.3.830 operator<<() [8/16]	279
7.1.3.831 operator<<() [9/16]	279
7.1.3.832 operator<<() [10/16]	280
7.1.3.833 operator<<() [11/16]	280
7.1.3.834 operator<<() [12/16]	280
7.1.3.835 operator<<() [13/16]	280
7.1.3.836 operator<<() [14/16]	281
7.1.3.837 operator<<() [15/16]	281
7.1.3.838 operator<<() [16/16]	281
7.1.3.839 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [34/34]	281
7.1.3.840 GINAC_BIND_UNARCHIVER() [50/50]	282
7.1.3.841 haswild()	282
7.1.3.842 GINAC_DECLARE_UNARCHIVER() [52/52]	282
7.1.3.843 wild()	282
7.1.4 Variable Documentation	282
7.1.4.1 unarch_table_instance	282
7.1.4.2 map_evalm	282
7.1.4.3 map_eval_integ	282
7.1.4.4 class_info< OPT >::first	283
7.1.4.5 class_info< OPT >::parents_identified	283
7.1.4.6 tensor	283
7.1.4.7 Pi	283
7.1.4.8 Euler	283

7.1.4.9 Catalan	284
7.1.4.10 crctab	284
7.1.4.11 library_initializer	284
7.1.4.12 _num0_bp	284
7.1.4.13 idx	284
7.1.4.14 force_include_tgamma	284
7.1.4.15 force_include_zeta1	284
7.1.4.16 GINAC_BIND_UNARCHIVER	285
7.1.4.17 l	285
7.1.4.18 Digits	285
7.1.4.19 next_print_context_id	285
7.1.4.20 structure< T, CP >::reg_info	286
7.1.4.21 version_major	286
7.1.4.22 version_minor	286
7.1.4.23 version_micro	286
7.1.4.24 _num_120_p	286
7.1.4.25 _ex_120	286
7.1.4.26 _num_60_p	286
7.1.4.27 _ex_60	286
7.1.4.28 _num_48_p	287
7.1.4.29 _ex_48	287
7.1.4.30 _num_30_p	287
7.1.4.31 _ex_30	287
7.1.4.32 _num_25_p	287
7.1.4.33 _ex_25	287
7.1.4.34 _num_24_p	287
7.1.4.35 _ex_24	287
7.1.4.36 _num_20_p	288
7.1.4.37 _ex_20	288
7.1.4.38 _num_18_p	288
7.1.4.39 _ex_18	288
7.1.4.40 _num_15_p	288
7.1.4.41 _ex_15	288
7.1.4.42 _num_12_p	288
7.1.4.43 _ex_12	288
7.1.4.44 _num_11_p	289
7.1.4.45 _ex_11	289
7.1.4.46 _num_10_p	289
7.1.4.47 _ex_10	289
7.1.4.48 _num_9_p	289
7.1.4.49 _ex_9	289
7.1.4.50 _num_8_p	289

7.1.4.51 _ex_8 . . . . .	289
7.1.4.52 _num_7_p . . . . .	290
7.1.4.53 _ex_7 . . . . .	290
7.1.4.54 _num_6_p . . . . .	290
7.1.4.55 _ex_6 . . . . .	290
7.1.4.56 _num_5_p . . . . .	290
7.1.4.57 _ex_5 . . . . .	290
7.1.4.58 _num_4_p . . . . .	290
7.1.4.59 _ex_4 . . . . .	290
7.1.4.60 _num_3_p . . . . .	291
7.1.4.61 _ex_3 . . . . .	291
7.1.4.62 _num_2_p . . . . .	291
7.1.4.63 _ex_2 . . . . .	291
7.1.4.64 _num_1_p . . . . .	291
7.1.4.65 _ex_1 . . . . .	291
7.1.4.66 _num_1_2_p . . . . .	292
7.1.4.67 _ex_1_2 . . . . .	292
7.1.4.68 _num_1_3_p . . . . .	292
7.1.4.69 _ex_1_3 . . . . .	292
7.1.4.70 _num_1_4_p . . . . .	292
7.1.4.71 _ex_1_4 . . . . .	292
7.1.4.72 _num0_p . . . . .	292
7.1.4.73 _ex0 . . . . .	293
7.1.4.74 _num1_4_p . . . . .	293
7.1.4.75 _ex1_4 . . . . .	293
7.1.4.76 _num1_3_p . . . . .	293
7.1.4.77 _ex1_3 . . . . .	293
7.1.4.78 _num1_2_p . . . . .	294
7.1.4.79 _ex1_2 . . . . .	294
7.1.4.80 _num1_p . . . . .	294
7.1.4.81 _ex1 . . . . .	294
7.1.4.82 _num2_p . . . . .	295
7.1.4.83 _ex2 . . . . .	295
7.1.4.84 _num3_p . . . . .	295
7.1.4.85 _ex3 . . . . .	295
7.1.4.86 _num4_p . . . . .	295
7.1.4.87 _ex4 . . . . .	296
7.1.4.88 _num5_p . . . . .	296
7.1.4.89 _ex5 . . . . .	296
7.1.4.90 _num6_p . . . . .	296
7.1.4.91 _ex6 . . . . .	296
7.1.4.92 _num7_p . . . . .	296

7.1.4.93 _ex7 . . . . .	296
7.1.4.94 _num8_p . . . . .	297
7.1.4.95 _ex8 . . . . .	297
7.1.4.96 _num9_p . . . . .	297
7.1.4.97 _ex9 . . . . .	297
7.1.4.98 _num10_p . . . . .	297
7.1.4.99 _ex10 . . . . .	297
7.1.4.100 _num11_p . . . . .	297
7.1.4.101 _ex11 . . . . .	298
7.1.4.102 _num12_p . . . . .	298
7.1.4.103 _ex12 . . . . .	298
7.1.4.104 _num15_p . . . . .	298
7.1.4.105 _ex15 . . . . .	298
7.1.4.106 _num18_p . . . . .	298
7.1.4.107 _ex18 . . . . .	298
7.1.4.108 _num20_p . . . . .	299
7.1.4.109 _ex20 . . . . .	299
7.1.4.110 _num24_p . . . . .	299
7.1.4.111 _ex24 . . . . .	299
7.1.4.112 _num25_p . . . . .	299
7.1.4.113 _ex25 . . . . .	299
7.1.4.114 _num30_p . . . . .	299
7.1.4.115 _ex30 . . . . .	299
7.1.4.116 _num48_p . . . . .	300
7.1.4.117 _ex48 . . . . .	300
7.1.4.118 _num60_p . . . . .	300
7.1.4.119 _ex60 . . . . .	300
7.1.4.120 _num120_p . . . . .	300
7.1.4.121 _ex120 . . . . .	300
7.2 GiNaC::Gt_detail Namespace Reference . . . . .	300
7.2.1 Function Documentation . . . . .	301
7.2.1.1 integer_partition() . . . . .	301
7.2.1.2 deconcatenate_path() . . . . .	301
7.2.1.3 deconcatenate_path< ex >() . . . . .	301
7.2.1.4 deconcatenate_path< Gt::kernel >() . . . . .	302
7.2.1.5 operator<<() . . . . .	302
7.2.1.6 operator<() . . . . .	302
7.3 GiNaC::internal Namespace Reference . . . . .	302
7.4 std Namespace Reference . . . . .	302
7.4.1 Function Documentation . . . . .	303
7.4.1.1 swap() . . . . .	303

<b>8 Class Documentation</b>	<b>305</b>
8.1 GiNaC::internal::_iter_rep Struct Reference	305
8.1.1 Constructor & Destructor Documentation	306
8.1.1.1 _iter_rep()	306
8.1.2 Member Function Documentation	306
8.1.2.1 operator==()	306
8.1.2.2 operator!=(())	306
8.1.3 Member Data Documentation	306
8.1.3.1 e	306
8.1.3.2 i	307
8.1.3.3 i_end	307
8.2 GiNaC::_numeric_digits Class Reference	307
8.2.1 Detailed Description	308
8.2.2 Constructor & Destructor Documentation	308
8.2.2.1 _numeric_digits()	308
8.2.3 Member Function Documentation	308
8.2.3.1 operator=()	308
8.2.3.2 operator long()	308
8.2.3.3 print()	308
8.2.3.4 add_callback()	309
8.2.4 Member Data Documentation	309
8.2.4.1 digits	309
8.2.4.2 too_late	309
8.2.4.3 callbacklist	309
8.3 GiNaC::add Class Reference	310
8.3.1 Detailed Description	316
8.3.2 Constructor & Destructor Documentation	316
8.3.2.1 add() [1/6]	316
8.3.2.2 add() [2/6]	316
8.3.2.3 add() [3/6]	317
8.3.2.4 add() [4/6]	317
8.3.2.5 add() [5/6]	317
8.3.2.6 add() [6/6]	317
8.3.3 Member Function Documentation	317
8.3.3.1 precedence()	317
8.3.3.2 info()	318
8.3.3.3 is_polynomial()	318
8.3.3.4 degree()	318
8.3.3.5 ldegree()	318
8.3.3.6 coeff()	319
8.3.3.7 eval()	319
8.3.3.8 evalm()	319



8.3.3.9 series()	320
8.3.3.10 normal()	320
8.3.3.11 integer_content()	320
8.3.3.12 smod()	321
8.3.3.13 max_coefficient()	321
8.3.3.14 conjugate()	321
8.3.3.15 real_part()	322
8.3.3.16 imag_part()	322
8.3.3.17 get_free_indices()	322
8.3.3.18 eval_ncmul()	322
8.3.3.19 derivative()	322
8.3.3.20 return_type()	323
8.3.3.21 return_type_tinfo()	323
8.3.3.22 thisexpairseq() [1/2]	323
8.3.3.23 thisexpairseq() [2/2]	323
8.3.3.24 split_ex_to_pair()	324
8.3.3.25 combine_ex_with_coeff_to_pair()	324
8.3.3.26 combine_pair_with_coeff_to_pair()	324
8.3.3.27 recombine_pair_to_ex()	324
8.3.3.28 expand()	325
8.3.3.29 print_add()	325
8.3.3.30 do_print()	325
8.3.3.31 do_print_latex()	325
8.3.3.32 do_print_csrc()	326
8.3.3.33 do_print_python_repr()	326
8.3.4 Friends And Related Symbol Documentation	326
8.3.4.1 mul	326
8.3.4.2 power	326
8.4 GiNaC::archive Class Reference	327
8.4.1 Detailed Description	328
8.4.2 Member Typedef Documentation	328
8.4.2.1 inv_at_cit	328
8.4.3 Constructor & Destructor Documentation	328
8.4.3.1 archive() [1/3]	328
8.4.3.2 ~archive()	328
8.4.3.3 archive() [2/3]	329
8.4.3.4 archive() [3/3]	329
8.4.4 Member Function Documentation	329
8.4.4.1 archive_ex()	329
8.4.4.2 unarchive_ex() [1/3]	329
8.4.4.3 unarchive_ex() [2/3]	330
8.4.4.4 unarchive_ex() [3/3]	330

8.4.4.5 num_expressions()	331
8.4.4.6 get_top_node()	331
8.4.4.7 clear()	331
8.4.4.8 add_node()	331
8.4.4.9 get_node()	331
8.4.4.10 forget()	332
8.4.4.11 printraw()	332
8.4.4.12 atomize()	332
8.4.4.13 unatomize()	332
8.4.5 Friends And Related Symbol Documentation	332
8.4.5.1 operator<<	332
8.4.5.2 operator>>	333
8.4.6 Member Data Documentation	333
8.4.6.1 nodes	333
8.4.6.2 exprs	333
8.4.6.3 atoms	333
8.4.6.4 inverse_atoms	333
8.4.6.5 exprtable	334
8.5 GiNaC::archive_node Class Reference	334
8.5.1 Detailed Description	336
8.5.2 Member Typedef Documentation	336
8.5.2.1 propinfovector	336
8.5.2.2 archive_node_cit	336
8.5.3 Member Enumeration Documentation	336
8.5.3.1 property_type	336
8.5.4 Constructor & Destructor Documentation	337
8.5.4.1 archive_node() [1/2]	337
8.5.4.2 archive_node() [2/2]	337
8.5.5 Member Function Documentation	337
8.5.5.1 operator=()	337
8.5.5.2 add_bool()	337
8.5.5.3 add_unsigned()	338
8.5.5.4 add_string()	338
8.5.5.5 add_ex()	338
8.5.5.6 find_bool()	338
8.5.5.7 find_unsigned()	339
8.5.5.8 find_string()	339
8.5.5.9 find_first()	339
8.5.5.10 find_last()	339
8.5.5.11 find_property_range()	340
8.5.5.12 find_ex()	340
8.5.5.13 find_ex_by_loc()	340

8.5.5.14 find_ex_node()	341
8.5.5.15 get_properties()	341
8.5.5.16 unarchive()	341
8.5.5.17 has_same_ex_as()	341
8.5.5.18 has_ex()	341
8.5.5.19 get_ex()	342
8.5.5.20 forget()	342
8.5.5.21 printraw()	342
8.5.6 Friends And Related Symbol Documentation	342
8.5.6.1 operator<<	342
8.5.6.2 operator>>	342
8.5.7 Member Data Documentation	343
8.5.7.1 a	343
8.5.7.2 props	343
8.5.7.3 has_expression	343
8.5.7.4 e	343
8.6 GiNaC::archive_node::archive_node_cit_range Struct Reference	344
8.6.1 Member Data Documentation	345
8.6.1.1 begin	345
8.6.1.2 end	345
8.7 GiNaC::archive::archived_ex Struct Reference	345
8.7.1 Detailed Description	345
8.7.2 Constructor & Destructor Documentation	345
8.7.2.1 archived_ex() [1/2]	345
8.7.2.2 archived_ex() [2/2]	346
8.7.3 Member Data Documentation	346
8.7.3.1 name	346
8.7.3.2 root	346
8.8 GiNaC::basic Class Reference	346
8.8.1 Detailed Description	351
8.8.2 Constructor & Destructor Documentation	351
8.8.2.1 basic() [1/2]	351
8.8.2.2 ~basic()	351
8.8.2.3 basic() [2/2]	351
8.8.3 Member Function Documentation	352
8.8.3.1 operator=()	352
8.8.3.2 duplicate()	352
8.8.3.3 eval()	352
8.8.3.4 evalf()	352
8.8.3.5 evalm()	353
8.8.3.6 eval_integ()	353
8.8.3.7 eval_ncmul()	353

8.8.3.8 eval_indexed()	353
8.8.3.9 print()	353
8.8.3.10 dbgprint()	354
8.8.3.11 dbgprinttree()	354
8.8.3.12 precedence()	354
8.8.3.13 info()	355
8.8.3.14 nops()	355
8.8.3.15 op()	355
8.8.3.16 operator[]() [1/4]	356
8.8.3.17 operator[]() [2/4]	356
8.8.3.18 let_op()	356
8.8.3.19 operator[]() [3/4]	356
8.8.3.20 operator[]() [4/4]	356
8.8.3.21 has()	357
8.8.3.22 match()	357
8.8.3.23 match_same_type()	357
8.8.3.24 subs()	358
8.8.3.25 map()	358
8.8.3.26 accept()	358
8.8.3.27 is_polynomial()	358
8.8.3.28 degree()	359
8.8.3.29 ldegree()	359
8.8.3.30 coeff()	359
8.8.3.31 expand()	359
8.8.3.32 collect()	360
8.8.3.33 derivative()	361
8.8.3.34 series()	361
8.8.3.35 normal()	362
8.8.3.36 to_rational()	362
8.8.3.37 to_polynomial()	362
8.8.3.38 integer_content()	363
8.8.3.39 smod()	363
8.8.3.40 max_coefficient()	363
8.8.3.41 get_free_indices()	364
8.8.3.42 add_indexed()	364
8.8.3.43 scalar_mul_indexed()	364
8.8.3.44 contract_with()	365
8.8.3.45 return_type()	366
8.8.3.46 return_type_tinfo()	366
8.8.3.47 conjugate()	366
8.8.3.48 real_part()	366
8.8.3.49 imag_part()	367

8.8.3.50 <code>compare_same_type()</code>	367
8.8.3.51 <code>is_equal_same_type()</code>	367
8.8.3.52 <code>calchash()</code>	368
8.8.3.53 <code>print_dispatch()</code> [1/2]	368
8.8.3.54 <code>print_dispatch()</code> [2/2]	368
8.8.3.55 <code>archive()</code>	369
8.8.3.56 <code>read_archive()</code>	369
8.8.3.57 <code>subs_one_level()</code>	370
8.8.3.58 <code>diff()</code>	370
8.8.3.59 <code>compare()</code>	370
8.8.3.60 <code>is_equal()</code>	371
8.8.3.61 <code>hold()</code>	371
8.8.3.62 <code>gethash()</code>	372
8.8.3.63 <code>setflag()</code>	372
8.8.3.64 <code>clearflag()</code>	372
8.8.3.65 <code>ensure_if_modifiable()</code>	373
8.8.3.66 <code>do_print()</code>	373
8.8.3.67 <code>do_print_tree()</code>	373
8.8.3.68 <code>do_print_python_repr()</code>	373
8.8.4 Friends And Related Symbol Documentation	374
8.8.4.1 <code>ex</code>	374
8.8.5 Member Data Documentation	375
8.8.5.1 <code>flags</code>	375
8.8.5.2 <code>hashvalue</code>	375
8.9 GiNaC::basic_log_kernel Class Reference	376
8.9.1 Detailed Description	381
8.9.2 Member Function Documentation	381
8.9.2.1 <code>series_coeff_impl()</code>	381
8.9.2.2 <code>do_print()</code>	381
8.10 GiNaC::basic_multi_iterator< T > Class Template Reference	382
8.10.1 Detailed Description	383
8.10.2 Constructor & Destructor Documentation	384
8.10.2.1 <code>basic_multi_iterator()</code> [1/3]	384
8.10.2.2 <code>basic_multi_iterator()</code> [2/3]	384
8.10.2.3 <code>basic_multi_iterator()</code> [3/3]	384
8.10.2.4 <code>~basic_multi_iterator()</code>	385
8.10.3 Member Function Documentation	385
8.10.3.1 <code>size()</code>	385
8.10.3.2 <code>overflow()</code>	385
8.10.3.3 <code>get_vector()</code>	385
8.10.3.4 <code>operator[]()</code> [1/2]	386
8.10.3.5 <code>operator[]()</code> [2/2]	386

8.10.3.6 operator>() [1/2]	386
8.10.3.7 operator>() [2/2]	386
8.10.3.8 init()	386
8.10.3.9 operator++()	387
8.10.4 Friends And Related Symbol Documentation	387
8.10.4.1 operator<<	387
8.10.5 Member Data Documentation	387
8.10.5.1 N	387
8.10.5.2 B	387
8.10.5.3 v	388
8.10.5.4 flag_overflow	388
8.11 GiNaC::basic_partition_generator Class Reference	388
8.11.1 Detailed Description	390
8.11.2 Constructor & Destructor Documentation	390
8.11.2.1 basic_partition_generator()	390
8.11.3 Member Data Documentation	390
8.11.3.1 mpgen	390
8.12 GiNaC::class_info< OPT > Class Template Reference	390
8.12.1 Constructor & Destructor Documentation	392
8.12.1.1 class_info()	392
8.12.2 Member Function Documentation	392
8.12.2.1 get_parent()	392
8.12.2.2 find()	392
8.12.2.3 dump_hierarchy()	392
8.12.2.4 dump_tree()	393
8.12.2.5 identify_parents()	393
8.12.3 Member Data Documentation	393
8.12.3.1 options	393
8.12.3.2 first	393
8.12.3.3 next	393
8.12.3.4 parent	393
8.12.3.5 parents_identified	394
8.13 GiNaC::clifford Class Reference	394
8.13.1 Detailed Description	401
8.13.2 Constructor & Destructor Documentation	401
8.13.2.1 clifford() [1/4]	401
8.13.2.2 clifford() [2/4]	402
8.13.2.3 clifford() [3/4]	402
8.13.2.4 clifford() [4/4]	402
8.13.3 Member Function Documentation	402
8.13.3.1 precedence()	402
8.13.3.2 archive()	403

8.13.3.3 read_archive()	403
8.13.3.4 eval_ncmul()	403
8.13.3.5 match_same_type()	404
8.13.3.6 thiscontainer() [1/2]	404
8.13.3.7 thiscontainer() [2/2]	404
8.13.3.8 return_type()	404
8.13.3.9 return_type_tinfo()	405
8.13.3.10 get_representation_label()	405
8.13.3.11 get_metric() [1/2]	405
8.13.3.12 get_metric() [2/2]	405
8.13.3.13 same_metric()	405
8.13.3.14 get_commutator_sign()	405
8.13.3.15 nops()	406
8.13.3.16 op()	406
8.13.3.17 let_op()	406
8.13.3.18 subs()	406
8.13.3.19 do_print_dflt()	407
8.13.3.20 do_print_latex()	407
8.13.3.21 do_print_tree()	407
8.13.4 Member Data Documentation	407
8.13.4.1 representation_label	407
8.13.4.2 metric	407
8.13.4.3 commutator_sign	408
8.14 GiNaC::cliffordunit Class Reference	408
8.14.1 Detailed Description	412
8.14.2 Member Function Documentation	413
8.14.2.1 contract_with()	413
8.14.2.2 do_print()	413
8.14.2.3 do_print_latex()	413
8.15 GiNaC::color Class Reference	413
8.15.1 Detailed Description	421
8.15.2 Constructor & Destructor Documentation	421
8.15.2.1 color() [1/4]	421
8.15.2.2 color() [2/4]	421
8.15.2.3 color() [3/4]	421
8.15.2.4 color() [4/4]	422
8.15.3 Member Function Documentation	422
8.15.3.1 archive()	422
8.15.3.2 read_archive()	422
8.15.3.3 eval_ncmul()	423
8.15.3.4 match_same_type()	423
8.15.3.5 thiscontainer() [1/2]	423

8.15.3.6 thiscontainer() [ 2 / 2 ]	424
8.15.3.7 return_type()	424
8.15.3.8 return_type_tinfo()	424
8.15.3.9 get_representation_label()	424
8.15.4 Member Data Documentation	424
8.15.4.1 representation_label	424
8.16 GiNaC::compare_all_equal< T > Class Template Reference	425
8.16.1 Detailed Description	425
8.16.2 Constructor & Destructor Documentation	425
8.16.2.1 ~compare_all_equal()	425
8.16.3 Member Function Documentation	425
8.16.3.1 struct_is_equal()	425
8.16.3.2 struct_compare()	425
8.17 GiNaC::compare_bitwise< T > Class Template Reference	426
8.17.1 Detailed Description	426
8.17.2 Constructor & Destructor Documentation	426
8.17.2.1 ~compare_bitwise()	426
8.17.3 Member Function Documentation	426
8.17.3.1 struct_is_equal()	426
8.17.3.2 struct_compare()	426
8.18 GiNaC::compare_std_less< T > Class Template Reference	427
8.18.1 Detailed Description	427
8.18.2 Constructor & Destructor Documentation	427
8.18.2.1 ~compare_std_less()	427
8.18.3 Member Function Documentation	427
8.18.3.1 struct_is_equal()	427
8.18.3.2 struct_compare()	427
8.19 GiNaC::composition_generator Class Reference	428
8.19.1 Detailed Description	429
8.19.2 Constructor & Destructor Documentation	429
8.19.2.1 composition_generator()	429
8.19.3 Member Function Documentation	429
8.19.3.1 get()	429
8.19.3.2 next()	429
8.19.4 Member Data Documentation	429
8.19.4.1 cmgen	429
8.19.4.2 atend	429
8.19.4.3 trivial	430
8.19.4.4 composition	430
8.19.4.5 current_updated	430
8.20 GiNaC::const_iterator Class Reference	430
8.20.1 Member Typedef Documentation	432



8.20.1.1 iterator_category	432
8.20.1.2 value_type	432
8.20.1.3 difference_type	432
8.20.1.4 pointer	432
8.20.1.5 reference	433
8.20.2 Constructor & Destructor Documentation	433
8.20.2.1 const_iterator() [1/2]	433
8.20.2.2 const_iterator() [2/2]	433
8.20.3 Member Function Documentation	433
8.20.3.1 operator*()	433
8.20.3.2 operator->()	433
8.20.3.3 operator[]()	433
8.20.3.4 operator++() [1/2]	433
8.20.3.5 operator++() [2/2]	434
8.20.3.6 operator+=()	434
8.20.3.7 operator+()	434
8.20.3.8 operator--() [1/2]	434
8.20.3.9 operator--() [2/2]	434
8.20.3.10 operator-=()	434
8.20.3.11 operator-()	434
8.20.3.12 operator==()	435
8.20.3.13 operator!=()	435
8.20.3.14 operator<()	435
8.20.3.15 operator>()	435
8.20.3.16 operator<=()	435
8.20.3.17 operator>=()	435
8.20.4 Friends And Related Symbol Documentation	435
8.20.4.1 ex	435
8.20.4.2 const_preorder_iterator	436
8.20.4.3 const_postorder_iterator	436
8.20.4.4 operator+	436
8.20.4.5 operator-	436
8.20.5 Member Data Documentation	436
8.20.5.1 e	436
8.20.5.2 i	436
8.21 GiNaC::const_postorder_iterator Class Reference	437
8.21.1 Member Typedef Documentation	437
8.21.1.1 iterator_category	437
8.21.1.2 value_type	437
8.21.1.3 difference_type	437
8.21.1.4 pointer	438
8.21.1.5 reference	438

8.21.2 Constructor & Destructor Documentation	438
8.21.2.1 const_postorder_iterator() [1/2]	438
8.21.2.2 const_postorder_iterator() [2/2]	438
8.21.3 Member Function Documentation	438
8.21.3.1 operator*()	438
8.21.3.2 operator->()	438
8.21.3.3 operator++() [1/2]	438
8.21.3.4 operator++() [2/2]	439
8.21.3.5 operator==()	439
8.21.3.6 operator!=(())	439
8.21.3.7 descend()	439
8.21.3.8 increment()	439
8.21.4 Member Data Documentation	439
8.21.4.1 s	439
8.22 GiNaC::const_preorder_iterator Class Reference	440
8.22.1 Member Typedef Documentation	440
8.22.1.1 iterator_category	440
8.22.1.2 value_type	440
8.22.1.3 difference_type	440
8.22.1.4 pointer	441
8.22.1.5 reference	441
8.22.2 Constructor & Destructor Documentation	441
8.22.2.1 const_preorder_iterator() [1/2]	441
8.22.2.2 const_preorder_iterator() [2/2]	441
8.22.3 Member Function Documentation	441
8.22.3.1 operator*()	441
8.22.3.2 operator->()	441
8.22.3.3 operator++() [1/2]	441
8.22.3.4 operator++() [2/2]	442
8.22.3.5 operator==()	442
8.22.3.6 operator!=(())	442
8.22.3.7 increment()	442
8.22.4 Member Data Documentation	442
8.22.4.1 s	442
8.23 GiNaC::constant Class Reference	443
8.23.1 Detailed Description	448
8.23.2 Constructor & Destructor Documentation	448
8.23.2.1 constant() [1/2]	448
8.23.2.2 constant() [2/2]	448
8.23.3 Member Function Documentation	449
8.23.3.1 info()	449
8.23.3.2 evalf()	449

8.23.3.3 is_polynomial()	449
8.23.3.4 conjugate()	449
8.23.3.5 real_part()	450
8.23.3.6 imag_part()	450
8.23.3.7 archive()	450
8.23.3.8 read_archive()	450
8.23.3.9 derivative()	451
8.23.3.10 is_equal_same_type()	451
8.23.3.11 calchash()	451
8.23.3.12 get_name()	451
8.23.3.13 get_TeX_name()	452
8.23.3.14 do_print()	452
8.23.3.15 do_print_tree()	452
8.23.3.16 do_print_latex()	452
8.23.3.17 do_print_python_repr()	452
8.23.4 Member Data Documentation	453
8.23.4.1 name	453
8.23.4.2 TeX_name	453
8.23.4.3 ef	453
8.23.4.4 number	453
8.23.4.5 serial	453
8.23.4.6 next_serial	453
8.23.4.7 domain	454
8.24 GiNaC::container< C > Class Template Reference	454
8.24.1 Detailed Description	459
8.24.2 Member Typedef Documentation	460
8.24.2.1 STLT	460
8.24.2.2 const_iterator	460
8.24.2.3 const_reverse_iterator	460
8.24.3 Constructor & Destructor Documentation	460
8.24.3.1 container() [1/4]	460
8.24.3.2 container() [2/4]	460
8.24.3.3 container() [3/4]	460
8.24.3.4 container() [4/4]	460
8.24.4 Member Function Documentation	461
8.24.4.1 get_default_flags()	461
8.24.4.2 get_open_delim()	461
8.24.4.3 get_close_delim()	461
8.24.4.4 info()	461
8.24.4.5 precedence()	461
8.24.4.6 nops()	462
8.24.4.7 op()	462

8.24.4.8 let_op()	462
8.24.4.9 subs()	463
8.24.4.10 read_archive()	463
8.24.4.11 archive()	463
8.24.4.12 conjugate()	464
8.24.4.13 real_part()	464
8.24.4.14 imag_part()	464
8.24.4.15 is_equal_same_type()	464
8.24.4.16 thiscontainer() [1/2]	465
8.24.4.17 thiscontainer() [2/2]	465
8.24.4.18 printseq()	465
8.24.4.19 sort_() [1/2]	465
8.24.4.20 sort_() [2/2]	466
8.24.4.21 unique_() [1/2]	466
8.24.4.22 prepend()	466
8.24.4.23 append()	466
8.24.4.24 remove_first()	466
8.24.4.25 remove_last()	467
8.24.4.26 remove_all()	467
8.24.4.27 sort()	467
8.24.4.28 unique()	467
8.24.4.29 begin()	467
8.24.4.30 end()	468
8.24.4.31 rbegin()	468
8.24.4.32 rend()	468
8.24.4.33 do_print()	468
8.24.4.34 do_print_tree()	468
8.24.4.35 do_print_python()	468
8.24.4.36 do_print_python_repr()	469
8.24.4.37 subchildren()	469
8.24.4.38 unique_() [2/2]	469
8.25 GiNaC::container_storage< C > Class Template Reference	469
8.25.1 Detailed Description	470
8.25.2 Member Typedef Documentation	470
8.25.2.1 STLT	470
8.25.3 Constructor & Destructor Documentation	470
8.25.3.1 container_storage() [1/4]	470
8.25.3.2 container_storage() [2/4]	471
8.25.3.3 container_storage() [3/4]	471
8.25.3.4 container_storage() [4/4]	471
8.25.3.5 ~container_storage()	471
8.25.4 Member Function Documentation	471

8.25.4.1 <code>reserve()</code> [1/4]	471
8.25.4.2 <code>reserve()</code> [2/4]	471
8.25.4.3 <code>reserve()</code> [3/4]	471
8.25.4.4 <code>reserve()</code> [4/4]	472
8.25.5 Member Data Documentation	472
8.25.5.1 <code>seq</code>	472
8.26 <code>GiNaC::composition_generator::coolmulti</code> Struct Reference	473
8.26.1 Constructor & Destructor Documentation	473
8.26.1.1 <code>coolmulti()</code>	473
8.26.1.2 <code>~coolmulti()</code>	474
8.26.2 Member Function Documentation	474
8.26.2.1 <code>next_permutation()</code>	474
8.26.2.2 <code>finished()</code>	474
8.26.3 Member Data Documentation	474
8.26.3.1 <code>head</code>	474
8.26.3.2 <code>i</code>	474
8.26.3.3 <code>after_i</code>	474
8.27 <code>GiNaC::derivative_map_function</code> Struct Reference	475
8.27.1 Detailed Description	476
8.27.2 Constructor & Destructor Documentation	476
8.27.2.1 <code>derivative_map_function()</code>	476
8.27.3 Member Function Documentation	476
8.27.3.1 <code>operator()()</code>	476
8.27.4 Member Data Documentation	476
8.27.4.1 <code>s</code>	476
8.28 <code>GiNaC::determinant_algo</code> Class Reference	477
8.28.1 Detailed Description	477
8.28.2 Member Enumeration Documentation	477
8.28.2.1 anonymous enum	477
8.29 <code>GiNaC::diracgamma</code> Class Reference	478
8.29.1 Detailed Description	483
8.29.2 Member Function Documentation	483
8.29.2.1 <code>contract_with()</code>	483
8.29.2.2 <code>do_print()</code>	483
8.29.2.3 <code>do_print_latex()</code>	483
8.30 <code>GiNaC::diracgamma5</code> Class Reference	484
8.30.1 Detailed Description	488
8.30.2 Member Function Documentation	488
8.30.2.1 <code>conjugate()</code>	488
8.30.2.2 <code>do_print()</code>	488
8.30.2.3 <code>do_print_latex()</code>	488
8.31 <code>GiNaC::diracgammaL</code> Class Reference	489

8.31.1 Detailed Description	493
8.31.2 Member Function Documentation	493
8.31.2.1 conjugate()	493
8.31.2.2 do_print()	493
8.31.2.3 do_print_latex()	493
8.32 GiNaC::diracgammaR Class Reference	494
8.32.1 Detailed Description	498
8.32.2 Member Function Documentation	498
8.32.2.1 conjugate()	498
8.32.2.2 do_print()	498
8.32.2.3 do_print_latex()	498
8.33 GiNaC::diracone Class Reference	499
8.33.1 Detailed Description	503
8.33.2 Member Function Documentation	503
8.33.2.1 do_print()	503
8.33.2.2 do_print_latex()	503
8.34 GiNaC::do_taylor Class Reference	504
8.34.1 Detailed Description	504
8.35 GiNaC::domain Class Reference	504
8.35.1 Detailed Description	504
8.35.2 Member Enumeration Documentation	504
8.35.2.1 anonymous enum	504
8.36 GiNaC::dunno Class Reference	505
8.36.1 Detailed Description	505
8.37 GiNaC::Ebar_kernel Class Reference	505
8.37.1 Detailed Description	510
8.37.2 Constructor & Destructor Documentation	511
8.37.2.1 Ebar_kernel()	511
8.37.3 Member Function Documentation	511
8.37.3.1 nops()	511
8.37.3.2 op()	511
8.37.3.3 let_op()	511
8.37.3.4 is_numeric()	512
8.37.3.5 get_numerical_value()	512
8.37.3.6 series_coeff_impl()	512
8.37.3.7 do_print()	512
8.37.4 Member Data Documentation	513
8.37.4.1 n	513
8.37.4.2 m	513
8.37.4.3 x	513
8.37.4.4 y	513
8.38 GiNaC::Eisenstein_h_kernel Class Reference	513

8.38.1 Detailed Description	519
8.38.2 Constructor & Destructor Documentation	520
8.38.2.1 Eisenstein_h_kernel()	520
8.38.3 Member Function Documentation	520
8.38.3.1 series()	520
8.38.3.2 nops()	520
8.38.3.3 op()	520
8.38.3.4 let_op()	521
8.38.3.5 is_numeric()	521
8.38.3.6 Laurent_series()	521
8.38.3.7 get_numerical_value()	521
8.38.3.8 uses_Laurent_series()	522
8.38.3.9 coefficient_a0()	522
8.38.3.10 coefficient_an()	522
8.38.3.11 q_expansion_modular_form()	522
8.38.3.12 do_print()	523
8.38.4 Member Data Documentation	523
8.38.4.1 k	523
8.38.4.2 N	523
8.38.4.3 r	523
8.38.4.4 s	523
8.38.4.5 C_norm	523
8.39 GiNaC::Eisenstein_kernel Class Reference	524
8.39.1 Detailed Description	530
8.39.2 Constructor & Destructor Documentation	530
8.39.2.1 Eisenstein_kernel()	530
8.39.3 Member Function Documentation	530
8.39.3.1 series()	530
8.39.3.2 nops()	531
8.39.3.3 op()	531
8.39.3.4 let_op()	531
8.39.3.5 is_numeric()	531
8.39.3.6 Laurent_series()	531
8.39.3.7 get_numerical_value()	532
8.39.3.8 uses_Laurent_series()	532
8.39.3.9 q_expansion_modular_form()	532
8.39.3.10 do_print()	532
8.39.4 Member Data Documentation	532
8.39.4.1 k	532
8.39.4.2 N	533
8.39.4.3 a	533
8.39.4.4 b	533

8.39.4.5 K	533
8.39.4.6 C_norm	533
8.40 GiNaC::composition_generator::coolmulti::element Struct Reference	533
8.40.1 Constructor & Destructor Documentation	534
8.40.1.1 element()	534
8.40.1.2 ~element()	534
8.40.2 Member Data Documentation	534
8.40.2.1 value	534
8.40.2.2 next	534
8.41 GiNaC::ELi_kernel Class Reference	535
8.41.1 Detailed Description	540
8.41.2 Constructor & Destructor Documentation	541
8.41.2.1 ELi_kernel()	541
8.41.3 Member Function Documentation	541
8.41.3.1 nops()	541
8.41.3.2 op()	541
8.41.3.3 let_op()	541
8.41.3.4 is_numeric()	542
8.41.3.5 get_numerical_value()	542
8.41.3.6 series_coeff_impl()	542
8.41.3.7 do_print()	542
8.41.4 Member Data Documentation	542
8.41.4.1 n	542
8.41.4.2 m	543
8.41.4.3 x	543
8.41.4.4 y	543
8.42 std::equal_to< GiNaC::ex > Struct Reference	543
8.42.1 Detailed Description	543
8.42.2 Member Function Documentation	543
8.42.2.1 operator()()	543
8.43 GiNaC::error_and_integral Struct Reference	544
8.43.1 Constructor & Destructor Documentation	545
8.43.1.1 error_and_integral()	545
8.43.2 Member Data Documentation	545
8.43.2.1 error	545
8.43.2.2 integral	545
8.44 GiNaC::error_and_integral_is_less Struct Reference	545
8.44.1 Member Function Documentation	545
8.44.1.1 operator()()	545
8.45 GiNaC::eval_integ_map_function Struct Reference	546
8.45.1 Detailed Description	547
8.45.2 Member Function Documentation	547



8.45.2.1 operator()	547
8.46 GiNaC::evalf_map_function Struct Reference	547
8.46.1 Detailed Description	548
8.46.2 Member Function Documentation	548
8.46.2.1 operator()	548
8.47 GiNaC::evalm_map_function Struct Reference	549
8.47.1 Detailed Description	550
8.47.2 Member Function Documentation	550
8.47.2.1 operator()	550
8.48 GiNaC::ex Class Reference	550
8.48.1 Detailed Description	554
8.48.2 Constructor & Destructor Documentation	554
8.48.2.1 ex() [1/10]	554
8.48.2.2 ex() [2/10]	554
8.48.2.3 ex() [3/10]	555
8.48.2.4 ex() [4/10]	555
8.48.2.5 ex() [5/10]	555
8.48.2.6 ex() [6/10]	555
8.48.2.7 ex() [7/10]	555
8.48.2.8 ex() [8/10]	555
8.48.2.9 ex() [9/10]	555
8.48.2.10 ex() [10/10]	556
8.48.3 Member Function Documentation	556
8.48.3.1 swap()	556
8.48.3.2 begin()	556
8.48.3.3 end()	556
8.48.3.4 preorder_begin()	556
8.48.3.5 preorder_end()	557
8.48.3.6 postorder_begin()	557
8.48.3.7 postorder_end()	557
8.48.3.8 eval()	557
8.48.3.9 evalf()	557
8.48.3.10 evalm()	557
8.48.3.11 eval_ncmul()	557
8.48.3.12 eval_integ()	558
8.48.3.13 print()	558
8.48.3.14 dbgprint()	558
8.48.3.15 dbgprinttree()	558
8.48.3.16 info()	559
8.48.3.17 nops()	559
8.48.3.18 op()	560
8.48.3.19 operator[]() [1/4]	560

8.48.3.20 operator[]() [2/4]	560
8.48.3.21 let_op()	560
8.48.3.22 operator[]() [3/4]	561
8.48.3.23 operator[]() [4/4]	561
8.48.3.24 lhs()	561
8.48.3.25 rhs()	561
8.48.3.26 conjugate()	561
8.48.3.27 real_part()	562
8.48.3.28 imag_part()	562
8.48.3.29 has()	562
8.48.3.30 find()	562
8.48.3.31 match() [1/2]	562
8.48.3.32 match() [2/2]	563
8.48.3.33 subs() [1/3]	563
8.48.3.34 subs() [2/3]	563
8.48.3.35 subs() [3/3]	563
8.48.3.36 map() [1/2]	564
8.48.3.37 map() [2/2]	564
8.48.3.38 accept()	564
8.48.3.39 traverse_preorder()	564
8.48.3.40 traverse_postorder()	564
8.48.3.41 traverse()	565
8.48.3.42 is_polynomial()	565
8.48.3.43 degree()	565
8.48.3.44 ldegree()	565
8.48.3.45 coeff()	565
8.48.3.46 lcoeff()	566
8.48.3.47 tcoeff()	566
8.48.3.48 expand()	566
8.48.3.49 collect()	566
8.48.3.50 diff()	566
8.48.3.51 series()	567
8.48.3.52 normal()	568
8.48.3.53 to_rational()	568
8.48.3.54 to_polynomial()	569
8.48.3.55 numer()	569
8.48.3.56 denom()	569
8.48.3.57 numer_denom()	570
8.48.3.58 unit()	570
8.48.3.59 content()	571
8.48.3.60 integer_content()	571
8.48.3.61 primpart() [1/2]	572

8.48.3.62	<a href="#">primpart()</a> [2/2]	572
8.48.3.63	<a href="#">unitcontprim()</a>	572
8.48.3.64	<a href="#">smod()</a>	573
8.48.3.65	<a href="#">max_coefficient()</a>	573
8.48.3.66	<a href="#">get_free_indices()</a>	573
8.48.3.67	<a href="#">simplify_indexed()</a> [1/2]	574
8.48.3.68	<a href="#">simplify_indexed()</a> [2/2]	574
8.48.3.69	<a href="#">compare()</a>	574
8.48.3.70	<a href="#">is_equal()</a>	575
8.48.3.71	<a href="#">is_zero()</a>	575
8.48.3.72	<a href="#">is_zero_matrix()</a>	575
8.48.3.73	<a href="#">symmetrize()</a> [1/2]	576
8.48.3.74	<a href="#">symmetrize()</a> [2/2]	576
8.48.3.75	<a href="#">antisymmetrize()</a> [1/2]	576
8.48.3.76	<a href="#">antisymmetrize()</a> [2/2]	576
8.48.3.77	<a href="#">symmetrize_cyclic()</a> [1/2]	576
8.48.3.78	<a href="#">symmetrize_cyclic()</a> [2/2]	577
8.48.3.79	<a href="#">return_type()</a>	577
8.48.3.80	<a href="#">return_type_tinfo()</a>	577
8.48.3.81	<a href="#">gethash()</a>	577
8.48.3.82	<a href="#">construct_from_basic()</a>	577
8.48.3.83	<a href="#">construct_from_int()</a>	578
8.48.3.84	<a href="#">construct_from_uint()</a>	578
8.48.3.85	<a href="#">construct_from_long()</a>	578
8.48.3.86	<a href="#">construct_from_ulong()</a>	578
8.48.3.87	<a href="#">construct_from_longlong()</a>	579
8.48.3.88	<a href="#">construct_from_ulonglong()</a>	579
8.48.3.89	<a href="#">construct_from_double()</a>	579
8.48.3.90	<a href="#">construct_from_string_and_lst()</a>	579
8.48.3.91	<a href="#">makewritable()</a>	579
8.48.3.92	<a href="#">share()</a>	580
8.48.4	<a href="#">Friends And Related Symbol Documentation</a>	580
8.48.4.1	<a href="#">archive_node</a>	580
8.48.4.2	<a href="#">are_ex_trivially_equal</a>	580
8.48.4.3	<a href="#">ex_to</a>	581
8.48.4.4	<a href="#">is_a</a>	581
8.48.4.5	<a href="#">is_exactly_a</a>	581
8.48.5	<a href="#">Member Data Documentation</a>	582
8.48.5.1	<a href="#">bp</a>	582
8.49	<a href="#">GiNaC::ex_base_is_less Struct Reference</a>	582
8.49.1	<a href="#">Member Function Documentation</a>	582
8.49.1.1	<a href="#">operator()()</a>	582

8.50 GiNaC::ex_is_equal Struct Reference	582
8.50.1 Member Function Documentation	583
8.50.1.1 operator()	583
8.51 GiNaC::ex_is_less Struct Reference	583
8.51.1 Member Function Documentation	583
8.51.1.1 operator()	583
8.52 GiNaC::ex_swap Struct Reference	583
8.52.1 Member Function Documentation	584
8.52.1.1 operator()	584
8.53 GiNaC::expair Class Reference	584
8.53.1 Detailed Description	586
8.53.2 Constructor & Destructor Documentation	586
8.53.2.1 expair() [1/2]	586
8.53.2.2 expair() [2/2]	586
8.53.3 Member Function Documentation	586
8.53.3.1 is_equal()	586
8.53.3.2 is_less()	587
8.53.3.3 compare()	587
8.53.3.4 print()	587
8.53.3.5 is_canonical_numeric()	587
8.53.3.6 swap()	587
8.53.3.7 conjugate()	588
8.53.4 Member Data Documentation	588
8.53.4.1 rest	588
8.53.4.2 coeff	588
8.54 GiNaC::expair_is_less Struct Reference	588
8.54.1 Detailed Description	589
8.54.2 Member Function Documentation	589
8.54.2.1 operator()	589
8.55 GiNaC::expair_rest_is_less Struct Reference	589
8.55.1 Detailed Description	589
8.55.2 Member Function Documentation	589
8.55.2.1 operator()	589
8.56 GiNaC::expair_swap Struct Reference	590
8.56.1 Member Function Documentation	590
8.56.1.1 operator()	590
8.57 GiNaC::expairseq Class Reference	590
8.57.1 Detailed Description	595
8.57.2 Constructor & Destructor Documentation	595
8.57.2.1 expairseq() [1/4]	595
8.57.2.2 expairseq() [2/4]	596
8.57.2.3 expairseq() [3/4]	596

8.57.2.4 expairseq() [4/4]	596
8.57.3 Member Function Documentation	596
8.57.3.1 precedence()	596
8.57.3.2 info()	596
8.57.3.3 nops()	597
8.57.3.4 op()	597
8.57.3.5 map()	597
8.57.3.6 eval()	597
8.57.3.7 to_rational()	598
8.57.3.8 to_polynomial()	598
8.57.3.9 match()	598
8.57.3.10 subs()	598
8.57.3.11 conjugate()	599
8.57.3.12 archive()	599
8.57.3.13 read_archive()	599
8.57.3.14 is_equal_same_type()	600
8.57.3.15 return_type()	600
8.57.3.16 calchash()	600
8.57.3.17 expand()	600
8.57.3.18 thisexpairseq() [1/2]	601
8.57.3.19 thisexpairseq() [2/2]	601
8.57.3.20 printseq()	601
8.57.3.21 printpair()	601
8.57.3.22 split_ex_to_pair()	602
8.57.3.23 combine_ex_with_coeff_to_pair()	602
8.57.3.24 combine_pair_with_coeff_to_pair()	602
8.57.3.25 recombine_pair_to_ex()	602
8.57.3.26 expair_needs_further_processing()	603
8.57.3.27 default_overall_coeff()	603
8.57.3.28 combine_overall_coeff() [1/2]	603
8.57.3.29 combine_overall_coeff() [2/2]	603
8.57.3.30 can_make_flat()	603
8.57.3.31 do_print()	604
8.57.3.32 do_print_tree()	604
8.57.3.33 construct_from_2_ex()	604
8.57.3.34 construct_from_2_expairseq()	604
8.57.3.35 construct_from_expairseq_ex()	604
8.57.3.36 construct_from_exvector()	605
8.57.3.37 construct_from_epvector() [1/2]	605
8.57.3.38 construct_from_epvector() [2/2]	605
8.57.3.39 make_flat() [1/2]	605
8.57.3.40 make_flat() [2/2]	605

8.57.3.41 canonicalize()	606
8.57.3.42 combine_same_terms_sorted_seq()	606
8.57.3.43 is_canonical()	606
8.57.3.44 expandchildren()	606
8.57.3.45 evalchildren()	607
8.57.3.46 subschildren()	607
8.57.4 Member Data Documentation	607
8.57.4.1 seq	607
8.57.4.2 overall_coeff	608
8.58 GiNaC::expand_map_function Struct Reference	608
8.58.1 Detailed Description	609
8.58.2 Constructor & Destructor Documentation	609
8.58.2.1 expand_map_function()	609
8.58.3 Member Function Documentation	610
8.58.3.1 operator()()	610
8.58.4 Member Data Documentation	610
8.58.4.1 options	610
8.59 GiNaC::expand_options Class Reference	610
8.59.1 Detailed Description	610
8.59.2 Member Enumeration Documentation	610
8.59.2.1 anonymous enum	610
8.60 GiNaC::factor_options Class Reference	611
8.60.1 Detailed Description	611
8.60.2 Member Enumeration Documentation	611
8.60.2.1 anonymous enum	611
8.61 GiNaC::fail Class Reference	612
8.61.1 Member Function Documentation	616
8.61.1.1 return_type()	616
8.61.1.2 do_print()	616
8.62 GiNaC::fderivative Class Reference	616
8.62.1 Detailed Description	624
8.62.2 Constructor & Destructor Documentation	624
8.62.2.1 fderivative() [1/3]	624
8.62.2.2 fderivative() [2/3]	624
8.62.2.3 fderivative() [3/3]	625
8.62.3 Member Function Documentation	625
8.62.3.1 print()	625
8.62.3.2 eval()	625
8.62.3.3 series()	626
8.62.3.4 thiscontainer() [1/2]	626
8.62.3.5 thiscontainer() [2/2]	626
8.62.3.6 archive()	626

8.62.3.7 read_archive()	627
8.62.3.8 derivative()	627
8.62.3.9 is_equal_same_type()	627
8.62.3.10 match_same_type()	628
8.62.3.11 derivatives()	628
8.62.3.12 do_print()	628
8.62.3.13 do_print_latex()	629
8.62.3.14 do_print_csrc()	629
8.62.3.15 do_print_tree()	629
8.62.4 Member Data Documentation	629
8.62.4.1 parameter_set	629
8.63 GiNaC::function Class Reference	630
8.63.1 Detailed Description	637
8.63.2 Constructor & Destructor Documentation	637
8.63.2.1 function() [1/18]	637
8.63.2.2 function() [2/18]	637
8.63.2.3 function() [3/18]	637
8.63.2.4 function() [4/18]	637
8.63.2.5 function() [5/18]	638
8.63.2.6 function() [6/18]	638
8.63.2.7 function() [7/18]	638
8.63.2.8 function() [8/18]	638
8.63.2.9 function() [9/18]	639
8.63.2.10 function() [10/18]	639
8.63.2.11 function() [11/18]	639
8.63.2.12 function() [12/18]	640
8.63.2.13 function() [13/18]	640
8.63.2.14 function() [14/18]	640
8.63.2.15 function() [15/18]	641
8.63.2.16 function() [16/18]	641
8.63.2.17 function() [17/18]	641
8.63.2.18 function() [18/18]	641
8.63.3 Member Function Documentation	641
8.63.3.1 print()	641
8.63.3.2 precedence()	642
8.63.3.3 expand()	642
8.63.3.4 eval()	642
8.63.3.5 evalf()	643
8.63.3.6 eval_ncmul()	643
8.63.3.7 calchash()	643
8.63.3.8 series()	643
8.63.3.9 thiscontainer() [1/2]	644

8.63.3.10 thiscontainer() [2/2]	644
8.63.3.11 conjugate()	644
8.63.3.12 real_part()	644
8.63.3.13 imag_part()	645
8.63.3.14 archive()	645
8.63.3.15 read_archive()	645
8.63.3.16 info()	645
8.63.3.17 derivative()	646
8.63.3.18 is_equal_same_type()	646
8.63.3.19 match_same_type()	646
8.63.3.20 return_type()	647
8.63.3.21 return_type_tinfo()	647
8.63.3.22 pderivative()	647
8.63.3.23 expl_derivative()	647
8.63.3.24 registered_functions()	647
8.63.3.25 lookup_remember_table()	648
8.63.3.26 store_remember_table()	648
8.63.3.27 power()	648
8.63.3.28 register_new()	648
8.63.3.29 find_function()	648
8.63.3.30 get_registered_functions()	649
8.63.3.31 get_serial()	649
8.63.3.32 get_name()	649
8.63.4 Friends And Related Symbol Documentation	649
8.63.4.1 remember_table_entry	649
8.63.5 Member Data Documentation	649
8.63.5.1 current_serial	649
8.63.5.2 serial	650
8.64 GiNaC::function_options Class Reference	650
8.64.1 Constructor & Destructor Documentation	656
8.64.1.1 function_options() [1/3]	656
8.64.1.2 function_options() [2/3]	657
8.64.1.3 function_options() [3/3]	657
8.64.1.4 ~function_options()	657
8.64.2 Member Function Documentation	657
8.64.2.1 initialize()	657
8.64.2.2 dummy()	658
8.64.2.3 set_name()	658
8.64.2.4 latex_name()	658
8.64.2.5 eval_func() [1/15]	658
8.64.2.6 eval_func() [2/15]	658
8.64.2.7 eval_func() [3/15]	658



8.64.2.8 eval_func() [4/15]	659
8.64.2.9 eval_func() [5/15]	659
8.64.2.10 eval_func() [6/15]	659
8.64.2.11 eval_func() [7/15]	659
8.64.2.12 eval_func() [8/15]	659
8.64.2.13 eval_func() [9/15]	659
8.64.2.14 eval_func() [10/15]	659
8.64.2.15 eval_func() [11/15]	660
8.64.2.16 eval_func() [12/15]	660
8.64.2.17 eval_func() [13/15]	660
8.64.2.18 eval_func() [14/15]	660
8.64.2.19 evalf_func() [1/15]	660
8.64.2.20 evalf_func() [2/15]	660
8.64.2.21 evalf_func() [3/15]	660
8.64.2.22 evalf_func() [4/15]	661
8.64.2.23 evalf_func() [5/15]	661
8.64.2.24 evalf_func() [6/15]	661
8.64.2.25 evalf_func() [7/15]	661
8.64.2.26 evalf_func() [8/15]	661
8.64.2.27 evalf_func() [9/15]	661
8.64.2.28 evalf_func() [10/15]	661
8.64.2.29 evalf_func() [11/15]	662
8.64.2.30 evalf_func() [12/15]	662
8.64.2.31 evalf_func() [13/15]	662
8.64.2.32 evalf_func() [14/15]	662
8.64.2.33 conjugate_func() [1/15]	662
8.64.2.34 conjugate_func() [2/15]	662
8.64.2.35 conjugate_func() [3/15]	662
8.64.2.36 conjugate_func() [4/15]	663
8.64.2.37 conjugate_func() [5/15]	663
8.64.2.38 conjugate_func() [6/15]	663
8.64.2.39 conjugate_func() [7/15]	663
8.64.2.40 conjugate_func() [8/15]	663
8.64.2.41 conjugate_func() [9/15]	663
8.64.2.42 conjugate_func() [10/15]	663
8.64.2.43 conjugate_func() [11/15]	664
8.64.2.44 conjugate_func() [12/15]	664
8.64.2.45 conjugate_func() [13/15]	664
8.64.2.46 conjugate_func() [14/15]	664
8.64.2.47 real_part_func() [1/15]	664
8.64.2.48 real_part_func() [2/15]	664
8.64.2.49 real_part_func() [3/15]	664

8.64.2.50 <code>real_part_func()</code> [4/15]	665
8.64.2.51 <code>real_part_func()</code> [5/15]	665
8.64.2.52 <code>real_part_func()</code> [6/15]	665
8.64.2.53 <code>real_part_func()</code> [7/15]	665
8.64.2.54 <code>real_part_func()</code> [8/15]	665
8.64.2.55 <code>real_part_func()</code> [9/15]	665
8.64.2.56 <code>real_part_func()</code> [10/15]	665
8.64.2.57 <code>real_part_func()</code> [11/15]	666
8.64.2.58 <code>real_part_func()</code> [12/15]	666
8.64.2.59 <code>real_part_func()</code> [13/15]	666
8.64.2.60 <code>real_part_func()</code> [14/15]	666
8.64.2.61 <code>imag_part_func()</code> [1/15]	666
8.64.2.62 <code>imag_part_func()</code> [2/15]	666
8.64.2.63 <code>imag_part_func()</code> [3/15]	666
8.64.2.64 <code>imag_part_func()</code> [4/15]	667
8.64.2.65 <code>imag_part_func()</code> [5/15]	667
8.64.2.66 <code>imag_part_func()</code> [6/15]	667
8.64.2.67 <code>imag_part_func()</code> [7/15]	667
8.64.2.68 <code>imag_part_func()</code> [8/15]	667
8.64.2.69 <code>imag_part_func()</code> [9/15]	667
8.64.2.70 <code>imag_part_func()</code> [10/15]	667
8.64.2.71 <code>imag_part_func()</code> [11/15]	668
8.64.2.72 <code>imag_part_func()</code> [12/15]	668
8.64.2.73 <code>imag_part_func()</code> [13/15]	668
8.64.2.74 <code>imag_part_func()</code> [14/15]	668
8.64.2.75 <code>expand_func()</code> [1/15]	668
8.64.2.76 <code>expand_func()</code> [2/15]	668
8.64.2.77 <code>expand_func()</code> [3/15]	668
8.64.2.78 <code>expand_func()</code> [4/15]	669
8.64.2.79 <code>expand_func()</code> [5/15]	669
8.64.2.80 <code>expand_func()</code> [6/15]	669
8.64.2.81 <code>expand_func()</code> [7/15]	669
8.64.2.82 <code>expand_func()</code> [8/15]	669
8.64.2.83 <code>expand_func()</code> [9/15]	669
8.64.2.84 <code>expand_func()</code> [10/15]	669
8.64.2.85 <code>expand_func()</code> [11/15]	670
8.64.2.86 <code>expand_func()</code> [12/15]	670
8.64.2.87 <code>expand_func()</code> [13/15]	670
8.64.2.88 <code>expand_func()</code> [14/15]	670
8.64.2.89 <code>derivative_func()</code> [1/15]	670
8.64.2.90 <code>derivative_func()</code> [2/15]	670
8.64.2.91 <code>derivative_func()</code> [3/15]	670

8.64.2.92 derivative_func() [4/15]	671
8.64.2.93 derivative_func() [5/15]	671
8.64.2.94 derivative_func() [6/15]	671
8.64.2.95 derivative_func() [7/15]	671
8.64.2.96 derivative_func() [8/15]	671
8.64.2.97 derivative_func() [9/15]	671
8.64.2.98 derivative_func() [10/15]	671
8.64.2.99 derivative_func() [11/15]	672
8.64.2.100 derivative_func() [12/15]	672
8.64.2.101 derivative_func() [13/15]	672
8.64.2.102 derivative_func() [14/15]	672
8.64.2.103 expl_derivative_func() [1/15]	672
8.64.2.104 expl_derivative_func() [2/15]	672
8.64.2.105 expl_derivative_func() [3/15]	672
8.64.2.106 expl_derivative_func() [4/15]	673
8.64.2.107 expl_derivative_func() [5/15]	673
8.64.2.108 expl_derivative_func() [6/15]	673
8.64.2.109 expl_derivative_func() [7/15]	673
8.64.2.110 expl_derivative_func() [8/15]	673
8.64.2.111 expl_derivative_func() [9/15]	673
8.64.2.112 expl_derivative_func() [10/15]	673
8.64.2.113 expl_derivative_func() [11/15]	674
8.64.2.114 expl_derivative_func() [12/15]	674
8.64.2.115 expl_derivative_func() [13/15]	674
8.64.2.116 expl_derivative_func() [14/15]	674
8.64.2.117 power_func() [1/15]	674
8.64.2.118 power_func() [2/15]	674
8.64.2.119 power_func() [3/15]	674
8.64.2.120 power_func() [4/15]	675
8.64.2.121 power_func() [5/15]	675
8.64.2.122 power_func() [6/15]	675
8.64.2.123 power_func() [7/15]	675
8.64.2.124 power_func() [8/15]	675
8.64.2.125 power_func() [9/15]	675
8.64.2.126 power_func() [10/15]	675
8.64.2.127 power_func() [11/15]	676
8.64.2.128 power_func() [12/15]	676
8.64.2.129 power_func() [13/15]	676
8.64.2.130 power_func() [14/15]	676
8.64.2.131 series_func() [1/15]	676
8.64.2.132 series_func() [2/15]	676
8.64.2.133 series_func() [3/15]	676

8.64.2.134 series_func() [4/15]	677
8.64.2.135 series_func() [5/15]	677
8.64.2.136 series_func() [6/15]	677
8.64.2.137 series_func() [7/15]	677
8.64.2.138 series_func() [8/15]	677
8.64.2.139 series_func() [9/15]	677
8.64.2.140 series_func() [10/15]	677
8.64.2.141 series_func() [11/15]	678
8.64.2.142 series_func() [12/15]	678
8.64.2.143 series_func() [13/15]	678
8.64.2.144 series_func() [14/15]	678
8.64.2.145 info_func() [1/15]	678
8.64.2.146 info_func() [2/15]	678
8.64.2.147 info_func() [3/15]	678
8.64.2.148 info_func() [4/15]	679
8.64.2.149 info_func() [5/15]	679
8.64.2.150 info_func() [6/15]	679
8.64.2.151 info_func() [7/15]	679
8.64.2.152 info_func() [8/15]	679
8.64.2.153 info_func() [9/15]	679
8.64.2.154 info_func() [10/15]	679
8.64.2.155 info_func() [11/15]	680
8.64.2.156 info_func() [12/15]	680
8.64.2.157 info_func() [13/15]	680
8.64.2.158 info_func() [14/15]	680
8.64.2.159 eval_func() [15/15]	680
8.64.2.160 evalf_func() [15/15]	680
8.64.2.161 conjugate_func() [15/15]	680
8.64.2.162 real_part_func() [15/15]	681
8.64.2.163 imag_part_func() [15/15]	681
8.64.2.164 expand_func() [15/15]	681
8.64.2.165 derivative_func() [15/15]	681
8.64.2.166 expl_derivative_func() [15/15]	681
8.64.2.167 power_func() [15/15]	681
8.64.2.168 series_func() [15/15]	681
8.64.2.169 info_func() [15/15]	682
8.64.2.170 print_func() [1/15]	682
8.64.2.171 print_func() [2/15]	682
8.64.2.172 print_func() [3/15]	682
8.64.2.173 print_func() [4/15]	682
8.64.2.174 print_func() [5/15]	682
8.64.2.175 print_func() [6/15]	683

8.64.2.176 print_func() [7/15]	683
8.64.2.177 print_func() [8/15]	683
8.64.2.178 print_func() [9/15]	683
8.64.2.179 print_func() [10/15]	683
8.64.2.180 print_func() [11/15]	683
8.64.2.181 print_func() [12/15]	684
8.64.2.182 print_func() [13/15]	684
8.64.2.183 print_func() [14/15]	684
8.64.2.184 print_func() [15/15]	684
8.64.2.185 set_return_type()	684
8.64.2.186 do_not_evalf_params()	684
8.64.2.187 remember()	685
8.64.2.188 overloaded()	685
8.64.2.189 set_symmetry()	685
8.64.2.190 get_name()	685
8.64.2.191 get_nparams()	685
8.64.2.192 has_derivative()	685
8.64.2.193 has_power()	685
8.64.2.194 test_and_set_nparams()	686
8.64.2.195 set_print_func()	686
8.64.3 Friends And Related Symbol Documentation	686
8.64.3.1 function	686
8.64.3.2 fderivative	687
8.64.4 Member Data Documentation	687
8.64.4.1 name	687
8.64.4.2 TeX_name	687
8.64.4.3 nparams	687
8.64.4.4 eval_f	687
8.64.4.5 evalf_f	687
8.64.4.6 conjugate_f	688
8.64.4.7 real_part_f	688
8.64.4.8 imag_part_f	688
8.64.4.9 expand_f	688
8.64.4.10 derivative_f	688
8.64.4.11 expl_derivative_f	688
8.64.4.12 power_f	689
8.64.4.13 series_f	689
8.64.4.14 print_dispatch_table	689
8.64.4.15 info_f	689
8.64.4.16 evalf_params_first	689
8.64.4.17 use_return_type	689
8.64.4.18 return_type	689

8.64.4.19	<a href="#">return_type_tinfo</a>	690
8.64.4.20	<a href="#">use_remember</a>	690
8.64.4.21	<a href="#">remember_size</a>	690
8.64.4.22	<a href="#">remember_assoc_size</a>	690
8.64.4.23	<a href="#">remember_strategy</a>	690
8.64.4.24	<a href="#">eval_use_exvector_args</a>	690
8.64.4.25	<a href="#">evalf_use_exvector_args</a>	690
8.64.4.26	<a href="#">conjugate_use_exvector_args</a>	690
8.64.4.27	<a href="#">real_part_use_exvector_args</a>	691
8.64.4.28	<a href="#">imag_part_use_exvector_args</a>	691
8.64.4.29	<a href="#">expand_use_exvector_args</a>	691
8.64.4.30	<a href="#">derivative_use_exvector_args</a>	691
8.64.4.31	<a href="#">expl_derivative_use_exvector_args</a>	691
8.64.4.32	<a href="#">power_use_exvector_args</a>	691
8.64.4.33	<a href="#">series_use_exvector_args</a>	691
8.64.4.34	<a href="#">print_use_exvector_args</a>	691
8.64.4.35	<a href="#">info_use_exvector_args</a>	692
8.64.4.36	<a href="#">functions_with_same_name</a>	692
8.64.4.37	<a href="#">symtree</a>	692
8.65	<a href="#">GiNaC::G2_SERIAL Class Reference</a>	692
8.65.1	<a href="#">Detailed Description</a>	692
8.65.2	<a href="#">Member Data Documentation</a>	692
8.65.2.1	<a href="#">serial</a>	692
8.66	<a href="#">GiNaC::G3_SERIAL Class Reference</a>	693
8.66.1	<a href="#">Detailed Description</a>	693
8.66.2	<a href="#">Member Data Documentation</a>	693
8.66.2.1	<a href="#">serial</a>	693
8.67	<a href="#">GiNaC::gcd_options Struct Reference</a>	693
8.67.1	<a href="#">Detailed Description</a>	693
8.67.2	<a href="#">Member Enumeration Documentation</a>	693
8.67.2.1	<a href="#">anonymous enum</a>	693
8.68	<a href="#">GiNaC::gcdheu_failed Class Reference</a>	694
8.68.1	<a href="#">Detailed Description</a>	694
8.69	<a href="#">GiNaC::Gt Class Reference</a>	694
8.69.1	<a href="#">Detailed Description</a>	699
8.69.2	<a href="#">Constructor &amp; Destructor Documentation</a>	699
8.69.2.1	<a href="#">Gt() [1/2]</a>	699
8.69.2.2	<a href="#">Gt() [2/2]</a>	700
8.69.3	<a href="#">Member Function Documentation</a>	700
8.69.3.1	<a href="#">setArgs()</a>	700
8.69.3.2	<a href="#">getTau()</a>	700
8.69.3.3	<a href="#">getZ()</a>	700

8.69.3.4	<a href="#">getArgs()</a>	700
8.69.3.5	<a href="#">precedence()</a>	700
8.69.3.6	<a href="#">eval()</a>	701
8.69.3.7	<a href="#">evalf()</a>	701
8.69.3.8	<a href="#">nops()</a>	701
8.69.3.9	<a href="#">subs()</a>	701
8.69.3.10	<a href="#">calchash()</a>	702
8.69.3.11	<a href="#">has()</a>	702
8.69.3.12	<a href="#">match()</a>	702
8.69.3.13	<a href="#">archive()</a>	702
8.69.3.14	<a href="#">read_archive()</a>	703
8.69.3.15	<a href="#">zisToFundamental()</a>	703
8.69.3.16	<a href="#">regularise()</a>	703
8.69.3.17	<a href="#">findMoebiusTransform()</a>	703
8.69.3.18	<a href="#">tauToFundamental()</a>	703
8.69.3.19	<a href="#">decomposeIntegrationPath()</a>	704
8.69.3.20	<a href="#">applyIntegrationPath()</a>	704
8.69.3.21	<a href="#">qExpand()</a>	704
8.69.3.22	<a href="#">evaluate()</a>	704
8.69.3.23	<a href="#">ex_zisToFundamental()</a> [1/2]	704
8.69.3.24	<a href="#">ex_regularise()</a> [1/2]	705
8.69.3.25	<a href="#">ex_tauToFundamental()</a> [1/2]	705
8.69.3.26	<a href="#">ex_cutIntegrationPath()</a> [1/2]	705
8.69.3.27	<a href="#">ex_qExpand()</a> [1/2]	705
8.69.3.28	<a href="#">ex_prepare()</a> [1/2]	705
8.69.3.29	<a href="#">ex_evaluate()</a> [1/2]	706
8.69.3.30	<a href="#">lst_evaluate()</a> [1/2]	706
8.69.3.31	<a href="#">ex_zisToFundamental()</a> [2/2]	706
8.69.3.32	<a href="#">ex_regularise()</a> [2/2]	706
8.69.3.33	<a href="#">ex_tauToFundamental()</a> [2/2]	706
8.69.3.34	<a href="#">ex_cutIntegrationPath()</a> [2/2]	707
8.69.3.35	<a href="#">ex_qExpand()</a> [2/2]	707
8.69.3.36	<a href="#">ex_prepare()</a> [2/2]	707
8.69.3.37	<a href="#">ex_evaluate()</a> [2/2]	707
8.69.3.38	<a href="#">lst_evaluate()</a> [2/2]	707
8.69.3.39	<a href="#">apply_function_recursive()</a>	707
8.69.3.40	<a href="#">clear_polylog_cache()</a>	708
8.69.3.41	<a href="#">do_print()</a>	708
8.69.3.42	<a href="#">do_print_latex()</a>	708
8.69.3.43	<a href="#">to_numeric()</a>	708
8.69.3.44	<a href="#">to_integer()</a>	708
8.69.4	<a href="#">Member Data Documentation</a>	709

8.69.4.1 cutIntegrationPath_threshold_real . . . . .	709
8.69.4.2 cutIntegrationPath_threshold_imag . . . . .	709
8.69.4.3 qExpand_minOrder . . . . .	709
8.69.4.4 qExpand_stepSize . . . . .	709
8.69.4.5 auto_clear_polylog_cache . . . . .	709
8.69.4.6 enable_tauToFundamental . . . . .	709
8.69.4.7 nargs . . . . .	709
8.69.4.8 args . . . . .	710
8.69.4.9 z . . . . .	710
8.69.4.10 tau . . . . .	710
8.69.4.11 ngexand_transformer . . . . .	710
8.70 GiNaC::has_distance< T > Class Template Reference . . . . .	710
8.70.1 Detailed Description . . . . .	711
8.70.2 Member Typedef Documentation . . . . .	711
8.70.2.1 yes_type . . . . .	711
8.70.2.2 no_type . . . . .	711
8.70.3 Member Enumeration Documentation . . . . .	711
8.70.3.1 anonymous enum . . . . .	711
8.70.4 Member Function Documentation . . . . .	712
8.70.4.1 test() [1/2] . . . . .	712
8.70.4.2 test() [2/2] . . . . .	712
8.71 GiNaC::has_options Class Reference . . . . .	712
8.71.1 Detailed Description . . . . .	712
8.71.2 Member Enumeration Documentation . . . . .	712
8.71.2.1 anonymous enum . . . . .	712
8.72 std::hash< GiNaC::ex > Struct Reference . . . . .	713
8.72.1 Detailed Description . . . . .	713
8.72.2 Member Function Documentation . . . . .	713
8.72.2.1 operator>() . . . . .	713
8.73 GiNaC::idx Class Reference . . . . .	713
8.73.1 Detailed Description . . . . .	719
8.73.2 Constructor & Destructor Documentation . . . . .	719
8.73.2.1 idx() . . . . .	719
8.73.3 Member Function Documentation . . . . .	720
8.73.3.1 info() . . . . .	720
8.73.3.2 nops() . . . . .	720
8.73.3.3 op() . . . . .	720
8.73.3.4 map() . . . . .	720
8.73.3.5 evalf() . . . . .	721
8.73.3.6 subs() . . . . .	721
8.73.3.7 archive() . . . . .	721
8.73.3.8 read_archive() . . . . .	722



8.73.3.9 derivative()	722
8.73.3.10 match_same_type()	722
8.73.3.11 calchash()	723
8.73.3.12 is_dummy_pair_same_type()	723
8.73.3.13 get_value()	723
8.73.3.14 is_numeric()	723
8.73.3.15 is_symbolic()	724
8.73.3.16 get_dim()	724
8.73.3.17 is_dim_numeric()	724
8.73.3.18 is_dim_symbolic()	724
8.73.3.19 replace_dim()	724
8.73.3.20 minimal_dim()	725
8.73.3.21 print_index()	725
8.73.3.22 do_print()	725
8.73.3.23 do_print_csrc()	725
8.73.3.24 do_print_latex()	725
8.73.3.25 do_print_tree()	726
8.73.4 Member Data Documentation	726
8.73.4.1 value	726
8.73.4.2 dim	726
8.74 GiNaC::idx_is_equal_ignore_dim Struct Reference	726
8.74.1 Member Function Documentation	727
8.74.1.1 operator()	727
8.75 GiNaC::indexed Class Reference	727
8.75.1 Detailed Description	733
8.75.2 Constructor & Destructor Documentation	734
8.75.2.1 indexed() [1/13]	734
8.75.2.2 indexed() [2/13]	734
8.75.2.3 indexed() [3/13]	734
8.75.2.4 indexed() [4/13]	735
8.75.2.5 indexed() [5/13]	735
8.75.2.6 indexed() [6/13]	736
8.75.2.7 indexed() [7/13]	736
8.75.2.8 indexed() [8/13]	737
8.75.2.9 indexed() [9/13]	738
8.75.2.10 indexed() [10/13]	738
8.75.2.11 indexed() [11/13]	739
8.75.2.12 indexed() [12/13]	739
8.75.2.13 indexed() [13/13]	739
8.75.3 Member Function Documentation	739
8.75.3.1 precedence()	739
8.75.3.2 info()	739

8.75.3.3 eval()	740
8.75.3.4 real_part()	740
8.75.3.5 imag_part()	740
8.75.3.6 get_free_indices()	740
8.75.3.7 archive()	740
8.75.3.8 read_archive()	741
8.75.3.9 derivative()	741
8.75.3.10 thiscontainer() [1/2]	741
8.75.3.11 thiscontainer() [2/2]	741
8.75.3.12 return_type()	742
8.75.3.13 return_type_tinfo()	742
8.75.3.14 expand()	742
8.75.3.15 all_index_values_are()	742
8.75.3.16 get_indices()	742
8.75.3.17 get_dummy_indices() [1/2]	743
8.75.3.18 get_dummy_indices() [2/2]	743
8.75.3.19 has_dummy_index_for()	743
8.75.3.20 get_symmetry()	743
8.75.3.21 printindices()	743
8.75.3.22 print_indexed()	744
8.75.3.23 do_print()	744
8.75.3.24 do_print_latex()	744
8.75.3.25 do_print_tree()	744
8.75.3.26 validate()	744
8.75.4 Friends And Related Symbol Documentation	745
8.75.4.1 simplify_indexed	745
8.75.4.2 simplify_indexed_product	745
8.75.4.3 reposition_dummy_indices	745
8.75.5 Member Data Documentation	746
8.75.5.1 symtree	746
8.76 GiNaC::info_flags Class Reference	746
8.76.1 Detailed Description	746
8.76.2 Member Enumeration Documentation	746
8.76.2.1 anonymous enum	746
8.77 GiNaC::integral Class Reference	747
8.77.1 Detailed Description	753
8.77.2 Constructor & Destructor Documentation	753
8.77.2.1 integral()	753
8.77.3 Member Function Documentation	753
8.77.3.1 precedence()	753
8.77.3.2 eval()	753
8.77.3.3 evalf()	754

8.77.3.4 degree()	754
8.77.3.5 ldegree()	754
8.77.3.6 eval_ncmul()	754
8.77.3.7 nops()	754
8.77.3.8 op()	755
8.77.3.9 let_op()	755
8.77.3.10 expand()	755
8.77.3.11 get_free_indices()	755
8.77.3.12 return_type()	756
8.77.3.13 return_type_tinfo()	756
8.77.3.14 conjugate()	756
8.77.3.15 eval_integ()	756
8.77.3.16 archive()	756
8.77.3.17 read_archive()	757
8.77.3.18 derivative()	757
8.77.3.19 series()	757
8.77.3.20 do_print()	758
8.77.3.21 do_print_latex()	758
8.77.4 Member Data Documentation	758
8.77.4.1 max_integration_level	758
8.77.4.2 relative_integration_error	758
8.77.4.3 x	758
8.77.4.4 a	758
8.77.4.5 b	759
8.77.4.6 f	759
8.78 GiNaC::integration_kernel Class Reference	759
8.78.1 Detailed Description	764
8.78.2 Member Function Documentation	764
8.78.2.1 series()	764
8.78.2.2 has_trailing_zero()	764
8.78.2.3 is_numeric()	765
8.78.2.4 Laurent_series()	765
8.78.2.5 get_numerical_value()	765
8.78.2.6 uses_Laurent_series()	766
8.78.2.7 series_coeff_impl()	766
8.78.2.8 get_cache_size()	766
8.78.2.9 set_cache_step()	766
8.78.2.10 get_series_coeff()	767
8.78.2.11 series_coeff()	767
8.78.2.12 get_numerical_value_impl()	767
8.78.2.13 do_print()	767
8.78.3 Member Data Documentation	768

8.78.3.1 cache_step_size . . . . .	768
8.78.3.2 series_vec . . . . .	768
8.79 GiNaC::is_not_a_clifford Struct Reference . . . . .	768
8.79.1 Detailed Description . . . . .	768
8.79.2 Member Function Documentation . . . . .	768
8.79.2.1 operator() . . . . .	768
8.80 GiNaC::is_summation_idx Struct Reference . . . . .	769
8.80.1 Member Function Documentation . . . . .	769
8.80.1.1 operator() . . . . .	769
8.81 GiNaC::iterated_integral2_SERIAL Class Reference . . . . .	769
8.81.1 Detailed Description . . . . .	769
8.81.2 Member Data Documentation . . . . .	769
8.81.2.1 serial . . . . .	769
8.82 GiNaC::iterated_integral3_SERIAL Class Reference . . . . .	770
8.82.1 Detailed Description . . . . .	770
8.82.2 Member Data Documentation . . . . .	770
8.82.2.1 serial . . . . .	770
8.83 GiNaC::Gt::kernel Struct Reference . . . . .	770
8.83.1 Constructor & Destructor Documentation . . . . .	772
8.83.1.1 kernel() [1/3] . . . . .	772
8.83.1.2 kernel() [2/3] . . . . .	772
8.83.1.3 kernel() [3/3] . . . . .	772
8.83.2 Member Function Documentation . . . . .	772
8.83.2.1 check() . . . . .	772
8.83.3 Friends And Related Symbol Documentation . . . . .	772
8.83.3.1 Gt . . . . .	772
8.83.4 Member Data Documentation . . . . .	773
8.83.4.1 ni . . . . .	773
8.83.4.2 zi . . . . .	773
8.83.4.3 deform . . . . .	773
8.84 GiNaC::Kronecker_dtau_kernel Class Reference . . . . .	773
8.84.1 Detailed Description . . . . .	778
8.84.2 Constructor & Destructor Documentation . . . . .	779
8.84.2.1 Kronecker_dtau_kernel() . . . . .	779
8.84.3 Member Function Documentation . . . . .	779
8.84.3.1 nops() . . . . .	779
8.84.3.2 op() . . . . .	779
8.84.3.3 let_op() . . . . .	779
8.84.3.4 is_numeric() . . . . .	780
8.84.3.5 get_numerical_value() . . . . .	780
8.84.3.6 series_coeff_impl() . . . . .	780
8.84.3.7 do_print() . . . . .	780

8.84.4 Member Data Documentation	781
8.84.4.1 n	781
8.84.4.2 z	781
8.84.4.3 K	781
8.84.4.4 C_norm	781
8.85 GiNaC::Kronecker_dz_kernel Class Reference	782
8.85.1 Detailed Description	787
8.85.2 Constructor & Destructor Documentation	788
8.85.2.1 Kronecker_dz_kernel()	788
8.85.3 Member Function Documentation	788
8.85.3.1 nops()	788
8.85.3.2 op()	788
8.85.3.3 let_op()	788
8.85.3.4 is_numeric()	789
8.85.3.5 get_numerical_value()	789
8.85.3.6 series_coeff_impl()	789
8.85.3.7 do_print()	789
8.85.4 Member Data Documentation	790
8.85.4.1 n	790
8.85.4.2 z_j	790
8.85.4.3 tau	790
8.85.4.4 K	790
8.85.4.5 C_norm	790
8.86 GiNaC::lanczos_coeffs Class Reference	790
8.86.1 Constructor & Destructor Documentation	791
8.86.1.1 lanczos_coeffs()	791
8.86.2 Member Function Documentation	791
8.86.2.1 sufficiently_accurate()	791
8.86.2.2 get_order()	791
8.86.2.3 calc_lanczos_A()	791
8.86.3 Member Data Documentation	792
8.86.3.1 coeffs	792
8.86.3.2 current_vector	792
8.87 std::less< GiNaC::ptr< T > > Struct Template Reference	792
8.87.1 Detailed Description	792
8.87.2 Member Function Documentation	792
8.87.2.1 operator()	792
8.88 GiNaC::library_init Class Reference	793
8.88.1 Detailed Description	793
8.88.2 Constructor & Destructor Documentation	794
8.88.2.1 library_init()	794
8.88.2.2 ~library_init()	794

8.88.3 Member Function Documentation	794
8.88.3.1 init_unarchivers()	794
8.88.4 Member Data Documentation	795
8.88.4.1 count	795
8.89 GiNaC::make_flat_inserter Class Reference	795
8.89.1 Detailed Description	795
8.89.2 Constructor & Destructor Documentation	796
8.89.2.1 make_flat_inserter() [1/2]	796
8.89.2.2 make_flat_inserter() [2/2]	796
8.89.3 Member Function Documentation	796
8.89.3.1 handle_factor()	796
8.89.3.2 combine_indices()	796
8.89.4 Member Data Documentation	796
8.89.4.1 do_renaming	796
8.89.4.2 used_indices	797
8.90 GiNaC::map_function Struct Reference	797
8.90.1 Detailed Description	799
8.90.2 Member Typedef Documentation	799
8.90.2.1 argument_type	799
8.90.2.2 result_type	799
8.90.3 Constructor & Destructor Documentation	799
8.90.3.1 ~map_function()	799
8.90.4 Member Function Documentation	799
8.90.4.1 operator()	799
8.91 GiNaC::matrix Class Reference	800
8.91.1 Detailed Description	805
8.91.2 Constructor & Destructor Documentation	805
8.91.2.1 matrix() [1/5]	805
8.91.2.2 matrix() [2/5]	806
8.91.2.3 matrix() [3/5]	806
8.91.2.4 matrix() [4/5]	806
8.91.2.5 matrix() [5/5]	806
8.91.3 Member Function Documentation	807
8.91.3.1 nops()	807
8.91.3.2 op()	807
8.91.3.3 let_op()	807
8.91.3.4 evalm()	807
8.91.3.5 subs()	808
8.91.3.6 eval_indexed()	808
8.91.3.7 add_indexed()	808
8.91.3.8 scalar_mul_indexed()	808
8.91.3.9 contract_with()	809

8.91.3.10	conjugate()	809
8.91.3.11	real_part()	809
8.91.3.12	imag_part()	809
8.91.3.13	archive()	809
8.91.3.14	read_archive()	810
8.91.3.15	match_same_type()	810
8.91.3.16	return_type()	810
8.91.3.17	rows()	810
8.91.3.18	cols()	811
8.91.3.19	add()	811
8.91.3.20	sub()	811
8.91.3.21	mul() [1/2]	811
8.91.3.22	mul() [2/2]	812
8.91.3.23	mul_scalar()	812
8.91.3.24	pow()	812
8.91.3.25	operator() [1/2]	812
8.91.3.26	operator() [2/2]	813
8.91.3.27	set()	814
8.91.3.28	transpose()	814
8.91.3.29	determinant()	814
8.91.3.30	trace()	815
8.91.3.31	charpoly()	816
8.91.3.32	inverse() [1/2]	816
8.91.3.33	inverse() [2/2]	816
8.91.3.34	solve()	817
8.91.3.35	rank() [1/2]	817
8.91.3.36	rank() [2/2]	818
8.91.3.37	is_zero_matrix()	818
8.91.3.38	determinant_minor()	818
8.91.3.39	echelon_form()	818
8.91.3.40	gauss_elimination()	819
8.91.3.41	division_free_elimination()	819
8.91.3.42	fraction_free_elimination()	820
8.91.3.43	markowitz_elimination()	820
8.91.3.44	pivot()	820
8.91.3.45	print_elements()	821
8.91.3.46	do_print()	821
8.91.3.47	do_print_latex()	821
8.91.3.48	do_print_python_repr()	822
8.91.4	Member Data Documentation	822
8.91.4.1	row	822
8.91.4.2	col	822

8.91.4.3 m	822
8.92 GiNaC::minkmetric Class Reference	823
8.92.1 Detailed Description	828
8.92.2 Constructor & Destructor Documentation	828
8.92.2.1 minkmetric()	828
8.92.3 Member Function Documentation	828
8.92.3.1 info()	828
8.92.3.2 eval_indexed()	829
8.92.3.3 archive()	829
8.92.3.4 read_archive()	829
8.92.3.5 return_type()	829
8.92.3.6 do_print()	829
8.92.3.7 do_print_latex()	830
8.92.4 Member Data Documentation	830
8.92.4.1 pos_sig	830
8.93 GiNaC::modular_form_kernel Class Reference	830
8.93.1 Detailed Description	835
8.93.2 Constructor & Destructor Documentation	836
8.93.2.1 modular_form_kernel()	836
8.93.3 Member Function Documentation	836
8.93.3.1 series()	836
8.93.3.2 nops()	836
8.93.3.3 op()	836
8.93.3.4 let_op()	837
8.93.3.5 is_numeric()	837
8.93.3.6 Laurent_series()	837
8.93.3.7 get_numerical_value()	837
8.93.3.8 uses_Laurent_series()	838
8.93.3.9 q_expansion_modular_form()	838
8.93.3.10 do_print()	838
8.93.4 Member Data Documentation	838
8.93.4.1 k	838
8.93.4.2 P	838
8.93.4.3 C_norm	839
8.94 GiNaC::basic_partition_generator::mpartition2 Struct Reference	839
8.94.1 Constructor & Destructor Documentation	839
8.94.1.1 mpartition2()	839
8.94.2 Member Function Documentation	839
8.94.2.1 next_partition()	839
8.94.3 Member Data Documentation	840
8.94.3.1 x	840
8.94.3.2 n	840



8.94.3.3 m	840
8.95 GiNaC::mul Class Reference	840
8.95.1 Detailed Description	846
8.95.2 Constructor & Destructor Documentation	846
8.95.2.1 mul() [1/7]	846
8.95.2.2 mul() [2/7]	846
8.95.2.3 mul() [3/7]	846
8.95.2.4 mul() [4/7]	847
8.95.2.5 mul() [5/7]	847
8.95.2.6 mul() [6/7]	847
8.95.2.7 mul() [7/7]	847
8.95.3 Member Function Documentation	847
8.95.3.1 precedence()	847
8.95.3.2 info()	848
8.95.3.3 is_polynomial()	848
8.95.3.4 degree()	848
8.95.3.5 ldegree()	848
8.95.3.6 coeff()	849
8.95.3.7 has()	849
8.95.3.8 eval()	849
8.95.3.9 evalf()	850
8.95.3.10 real_part()	850
8.95.3.11 imag_part()	850
8.95.3.12 evalm()	850
8.95.3.13 series()	850
8.95.3.14 normal()	851
8.95.3.15 integer_content()	851
8.95.3.16 smod()	851
8.95.3.17 max_coefficient()	852
8.95.3.18 get_free_indices()	852
8.95.3.19 conjugate()	852
8.95.3.20 derivative()	852
8.95.3.21 eval_ncmul()	853
8.95.3.22 return_type()	853
8.95.3.23 return_type_tinfo()	853
8.95.3.24 thisexpairseq() [1/2]	853
8.95.3.25 thisexpairseq() [2/2]	853
8.95.3.26 split_ex_to_pair()	854
8.95.3.27 combine_ex_with_coeff_to_pair()	854
8.95.3.28 combine_pair_with_coeff_to_pair()	854
8.95.3.29 recombine_pair_to_ex()	854
8.95.3.30 expair_needs_further_processing()	855

8.95.3.31 default_overall_coeff()	855
8.95.3.32 combine_overall_coeff() [1/2]	855
8.95.3.33 combine_overall_coeff() [2/2]	855
8.95.3.34 can_make_flat()	855
8.95.3.35 expand()	856
8.95.3.36 algebraic_subs_mul()	856
8.95.3.37 find_real_imag()	856
8.95.3.38 print_overall_coeff()	856
8.95.3.39 do_print()	857
8.95.3.40 do_print_latex()	857
8.95.3.41 do_print_csrc()	857
8.95.3.42 do_print_python_repr()	857
8.95.3.43 can_be_further_expanded()	857
8.95.3.44 expandchildren()	858
8.95.4 Friends And Related Symbol Documentation	858
8.95.4.1 add	858
8.95.4.2 ncmul	858
8.95.4.3 power	858
8.96 GiNaC::multi_iterator_counter< T > Class Template Reference	859
8.96.1 Detailed Description	860
8.96.2 Constructor & Destructor Documentation	861
8.96.2.1 multi_iterator_counter() [1/3]	861
8.96.2.2 multi_iterator_counter() [2/3]	861
8.96.2.3 multi_iterator_counter() [3/3]	861
8.96.3 Member Function Documentation	861
8.96.3.1 init()	861
8.96.3.2 operator++()	862
8.96.4 Friends And Related Symbol Documentation	862
8.96.4.1 operator<<	862
8.97 GiNaC::multi_iterator_counter_indv< T > Class Template Reference	862
8.97.1 Detailed Description	864
8.97.2 Constructor & Destructor Documentation	864
8.97.2.1 multi_iterator_counter_indv() [1/3]	864
8.97.2.2 multi_iterator_counter_indv() [2/3]	865
8.97.2.3 multi_iterator_counter_indv() [3/3]	865
8.97.3 Member Function Documentation	865
8.97.3.1 init()	865
8.97.3.2 operator++()	865
8.97.4 Friends And Related Symbol Documentation	866
8.97.4.1 operator<<	866
8.97.5 Member Data Documentation	866
8.97.5.1 Nv	866

8.98 GiNaC::multi_iterator_ordered< T > Class Template Reference	866
8.98.1 Detailed Description	868
8.98.2 Constructor & Destructor Documentation	868
8.98.2.1 multi_iterator_ordered() [1/3]	868
8.98.2.2 multi_iterator_ordered() [2/3]	869
8.98.2.3 multi_iterator_ordered() [3/3]	869
8.98.3 Member Function Documentation	869
8.98.3.1 init()	869
8.98.3.2 operator++()	869
8.98.4 Friends And Related Symbol Documentation	870
8.98.4.1 operator<<	870
8.99 GiNaC::multi_iterator_ordered_eq< T > Class Template Reference	870
8.99.1 Detailed Description	872
8.99.2 Constructor & Destructor Documentation	872
8.99.2.1 multi_iterator_ordered_eq() [1/3]	872
8.99.2.2 multi_iterator_ordered_eq() [2/3]	872
8.99.2.3 multi_iterator_ordered_eq() [3/3]	872
8.99.3 Member Function Documentation	873
8.99.3.1 init()	873
8.99.3.2 operator++()	873
8.99.4 Friends And Related Symbol Documentation	873
8.99.4.1 operator<<	873
8.100 GiNaC::multi_iterator_ordered_eq_indv< T > Class Template Reference	874
8.100.1 Detailed Description	875
8.100.2 Constructor & Destructor Documentation	876
8.100.2.1 multi_iterator_ordered_eq_indv() [1/3]	876
8.100.2.2 multi_iterator_ordered_eq_indv() [2/3]	876
8.100.2.3 multi_iterator_ordered_eq_indv() [3/3]	876
8.100.3 Member Function Documentation	876
8.100.3.1 init()	876
8.100.3.2 operator++()	877
8.100.4 Friends And Related Symbol Documentation	877
8.100.4.1 operator<<	877
8.100.5 Member Data Documentation	877
8.100.5.1 Nv	877
8.101 GiNaC::multi_iterator_permutation< T > Class Template Reference	878
8.101.1 Detailed Description	880
8.101.2 Constructor & Destructor Documentation	880
8.101.2.1 multi_iterator_permutation() [1/3]	880
8.101.2.2 multi_iterator_permutation() [2/3]	880
8.101.2.3 multi_iterator_permutation() [3/3]	880
8.101.3 Member Function Documentation	881

8.101.3.1 init()	881
8.101.3.2 operator++()	881
8.101.3.3 get_sign()	881
8.101.4 Friends And Related Symbol Documentation	882
8.101.4.1 operator<<	882
8.102 GiNaC::multi_iterator_shuffle< T > Class Template Reference	882
8.102.1 Detailed Description	884
8.102.2 Constructor & Destructor Documentation	884
8.102.2.1 multi_iterator_shuffle() [1/2]	884
8.102.2.2 multi_iterator_shuffle() [2/2]	884
8.102.3 Member Function Documentation	885
8.102.3.1 init()	885
8.102.3.2 operator++()	885
8.102.4 Friends And Related Symbol Documentation	885
8.102.4.1 operator<<	885
8.102.5 Member Data Documentation	885
8.102.5.1 N_internal	885
8.102.5.2 v_internal	886
8.102.5.3 v_orig	886
8.103 GiNaC::multi_iterator_shuffle_prime< T > Class Template Reference	886
8.103.1 Detailed Description	888
8.103.2 Constructor & Destructor Documentation	889
8.103.2.1 multi_iterator_shuffle_prime() [1/2]	889
8.103.2.2 multi_iterator_shuffle_prime() [2/2]	889
8.103.3 Member Function Documentation	889
8.103.3.1 init()	889
8.103.4 Friends And Related Symbol Documentation	889
8.103.4.1 operator<<	889
8.104 GiNaC::multiple_polylog_kernel Class Reference	890
8.104.1 Detailed Description	895
8.104.2 Constructor & Destructor Documentation	896
8.104.2.1 multiple_polylog_kernel()	896
8.104.3 Member Function Documentation	896
8.104.3.1 nops()	896
8.104.3.2 op()	896
8.104.3.3 let_op()	896
8.104.3.4 is_numeric()	896
8.104.3.5 series_coeff_impl()	897
8.104.3.6 do_print()	897
8.104.4 Member Data Documentation	897
8.104.4.1 z	897
8.105 GiNaC::ncmul Class Reference	898

8.105.1 Detailed Description	903
8.105.2 Constructor & Destructor Documentation	904
8.105.2.1 ncmul() [1/7]	904
8.105.2.2 ncmul() [2/7]	904
8.105.2.3 ncmul() [3/7]	904
8.105.2.4 ncmul() [4/7]	904
8.105.2.5 ncmul() [5/7]	904
8.105.2.6 ncmul() [6/7]	904
8.105.2.7 ncmul() [7/7]	905
8.105.3 Member Function Documentation	905
8.105.3.1 precedence()	905
8.105.3.2 info()	905
8.105.3.3 degree()	905
8.105.3.4 ldegree()	905
8.105.3.5 expand()	906
8.105.3.6 coeff()	906
8.105.3.7 eval()	906
8.105.3.8 evalm()	907
8.105.3.9 get_free_indices()	907
8.105.3.10 thiscontainer() [1/2]	907
8.105.3.11 thiscontainer() [2/2]	907
8.105.3.12 conjugate()	908
8.105.3.13 real_part()	908
8.105.3.14 imag_part()	908
8.105.3.15 derivative()	908
8.105.3.16 return_type()	908
8.105.3.17 return_type_tinfo()	909
8.105.3.18 do_print()	909
8.105.3.19 do_print_csrc()	909
8.105.3.20 count_factors()	909
8.105.3.21 append_factors()	909
8.105.3.22 expandchildren()	910
8.105.3.23 get_factors()	910
8.105.4 Friends And Related Symbol Documentation	910
8.105.4.1 power	910
8.105.4.2 reeval_ncmul	910
8.105.4.3 hold_ncmul	910
8.106 GiNaC::normal_map_function Struct Reference	911
8.106.1 Detailed Description	912
8.106.2 Member Function Documentation	912
8.106.2.1 operator()()	912
8.107 GiNaC::numeric Class Reference	912

8.107.1 Detailed Description	918
8.107.2 Constructor & Destructor Documentation	919
8.107.2.1 numeric() [1/10]	919
8.107.2.2 numeric() [2/10]	919
8.107.2.3 numeric() [3/10]	919
8.107.2.4 numeric() [4/10]	919
8.107.2.5 numeric() [5/10]	919
8.107.2.6 numeric() [6/10]	919
8.107.2.7 numeric() [7/10]	920
8.107.2.8 numeric() [8/10]	920
8.107.2.9 numeric() [9/10]	920
8.107.2.10 numeric() [10/10]	920
8.107.3 Member Function Documentation	920
8.107.3.1 precedence()	920
8.107.3.2 info()	921
8.107.3.3 is_polynomial()	921
8.107.3.4 degree()	921
8.107.3.5 ldegree()	921
8.107.3.6 coeff()	922
8.107.3.7 has()	922
8.107.3.8 eval()	922
8.107.3.9 evalf()	922
8.107.3.10 subs()	923
8.107.3.11 normal()	923
8.107.3.12 to_rational()	923
8.107.3.13 to_polynomial()	924
8.107.3.14 integer_content()	924
8.107.3.15 smod()	924
8.107.3.16 max_coefficient()	925
8.107.3.17 conjugate()	925
8.107.3.18 real_part()	925
8.107.3.19 imag_part()	925
8.107.3.20 archive()	925
8.107.3.21 read_archive()	926
8.107.3.22 derivative()	926
8.107.3.23 is_equal_same_type()	926
8.107.3.24 calchash()	926
8.107.3.25 add()	927
8.107.3.26 sub()	927
8.107.3.27 mul()	927
8.107.3.28 div()	927
8.107.3.29 power()	928

8.107.3.30 add_dyn()	928
8.107.3.31 sub_dyn()	928
8.107.3.32 mul_dyn()	928
8.107.3.33 div_dyn()	929
8.107.3.34 power_dyn()	929
8.107.3.35 operator=() [1/6]	929
8.107.3.36 operator=() [2/6]	929
8.107.3.37 operator=() [3/6]	929
8.107.3.38 operator=() [4/6]	930
8.107.3.39 operator=() [5/6]	930
8.107.3.40 operator=() [6/6]	930
8.107.3.41 inverse()	930
8.107.3.42 step()	930
8.107.3.43 csgn()	931
8.107.3.44 compare()	931
8.107.3.45 is_equal()	931
8.107.3.46 is_zero()	932
8.107.3.47 is_positive()	932
8.107.3.48 is_negative()	932
8.107.3.49 is_integer()	932
8.107.3.50 is_pos_integer()	933
8.107.3.51 is_nonneg_integer()	933
8.107.3.52 is_even()	933
8.107.3.53 is_odd()	933
8.107.3.54 is_prime()	933
8.107.3.55 is_rational()	934
8.107.3.56 is_real()	934
8.107.3.57 is_cinteger()	934
8.107.3.58 is_crational()	934
8.107.3.59 operator==(())	934
8.107.3.60 operator!=(())	935
8.107.3.61 operator<()	935
8.107.3.62 operator<=()	935
8.107.3.63 operator>()	935
8.107.3.64 operator>=()	936
8.107.3.65 to_int()	936
8.107.3.66 to_long()	936
8.107.3.67 to_double()	936
8.107.3.68 to_cl_N()	937
8.107.3.69 real()	937
8.107.3.70 imag()	937
8.107.3.71 numer()	937

8.107.3.72	denom()	938
8.107.3.73	int_length()	938
8.107.3.74	print_numeric()	938
8.107.3.75	do_print()	938
8.107.3.76	do_print_latex()	939
8.107.3.77	do_print_csrc()	939
8.107.3.78	do_print_csrc_cl_N()	939
8.107.3.79	do_print_tree()	939
8.107.3.80	do_print_python_repr()	939
8.107.4	Member Data Documentation	940
8.107.4.1	value	940
8.108	GiNaC::op0_is_equal Struct Reference	940
8.108.1	Member Function Documentation	940
8.108.1.1	operator()	940
8.109	GiNaC::partition_generator Class Reference	941
8.109.1	Detailed Description	942
8.109.2	Constructor & Destructor Documentation	942
8.109.2.1	partition_generator()	942
8.109.3	Member Function Documentation	942
8.109.3.1	get()	942
8.109.3.2	next()	942
8.109.4	Member Data Documentation	943
8.109.4.1	partition	943
8.109.4.2	current_updated	943
8.110	GiNaC::partition_with_zero_parts_generator Class Reference	943
8.110.1	Detailed Description	944
8.110.2	Constructor & Destructor Documentation	945
8.110.2.1	partition_with_zero_parts_generator()	945
8.110.3	Member Function Documentation	945
8.110.3.1	get()	945
8.110.3.2	next()	945
8.110.4	Member Data Documentation	945
8.110.4.1	m	945
8.110.4.2	partition	945
8.110.4.3	current_updated	946
8.111	GiNaC::Gt_detail::pathintegral_term Class Reference	946
8.111.1	Constructor & Destructor Documentation	947
8.111.1.1	pathintegral_term() [1/2]	947
8.111.1.2	pathintegral_term() [2/2]	947
8.111.2	Member Function Documentation	948
8.111.2.1	integrate()	948
8.111.2.2	evaluate()	948



8.111.2.3 G_path() [1/2]	948
8.111.2.4 G_path() [2/2]	948
8.111.3 Friends And Related Symbol Documentation	948
8.111.3.1 pathintegral_term_list	948
8.111.3.2 operator<<	949
8.111.3.3 operator<	949
8.111.4 Member Data Documentation	949
8.111.4.1 prefactor	949
8.111.4.2 power	949
8.111.4.3 polylog	949
8.111.4.4 denom	949
8.112 GiNaC::Gt_detail::pathintegral_term_list Class Reference	950
8.112.1 Member Function Documentation	950
8.112.1.1 begin()	950
8.112.1.2 end()	950
8.112.1.3 clear()	950
8.112.1.4 empty()	951
8.112.1.5 size()	951
8.112.1.6 add() [1/3]	951
8.112.1.7 add() [2/3]	951
8.112.1.8 add() [3/3]	951
8.112.1.9 pop()	951
8.112.2 Member Data Documentation	952
8.112.2.1 terms	952
8.113 GiNaC::pointer_to_map_function Class Reference	952
8.113.1 Constructor & Destructor Documentation	954
8.113.1.1 pointer_to_map_function()	954
8.113.2 Member Function Documentation	954
8.113.2.1 operator>()	954
8.113.3 Member Data Documentation	954
8.113.3.1 ptr	954
8.114 GiNaC::pointer_to_map_function_1arg< T1 > Class Template Reference	954
8.114.1 Constructor & Destructor Documentation	956
8.114.1.1 pointer_to_map_function_1arg()	956
8.114.2 Member Function Documentation	956
8.114.2.1 operator>()	956
8.114.3 Member Data Documentation	956
8.114.3.1 ptr	956
8.114.3.2 arg1	956
8.115 GiNaC::pointer_to_map_function_2args< T1, T2 > Class Template Reference	957
8.115.1 Constructor & Destructor Documentation	958
8.115.1.1 pointer_to_map_function_2args()	958

8.115.2 Member Function Documentation	958
8.115.2.1 operator()()	958
8.115.3 Member Data Documentation	959
8.115.3.1 ptr	959
8.115.3.2 arg1	959
8.115.3.3 arg2	959
8.116 GiNaC::pointer_to_map_function_3args< T1, T2, T3 > Class Template Reference	959
8.116.1 Constructor & Destructor Documentation	961
8.116.1.1 pointer_to_map_function_3args()	961
8.116.2 Member Function Documentation	961
8.116.2.1 operator()()	961
8.116.3 Member Data Documentation	961
8.116.3.1 ptr	961
8.116.3.2 arg1	961
8.116.3.3 arg2	962
8.116.3.4 arg3	962
8.117 GiNaC::pointer_to_member_to_map_function< C > Class Template Reference	962
8.117.1 Constructor & Destructor Documentation	964
8.117.1.1 pointer_to_member_to_map_function()	964
8.117.2 Member Function Documentation	964
8.117.2.1 operator()()	964
8.117.3 Member Data Documentation	964
8.117.3.1 ptr	964
8.117.3.2 c	964
8.118 GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > Class Template Reference	965
8.118.1 Constructor & Destructor Documentation	967
8.118.1.1 pointer_to_member_to_map_function_1arg()	967
8.118.2 Member Function Documentation	967
8.118.2.1 operator()()	967
8.118.3 Member Data Documentation	967
8.118.3.1 ptr	967
8.118.3.2 c	967
8.118.3.3 arg1	968
8.119 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > Class Template Reference	968
8.119.1 Constructor & Destructor Documentation	970
8.119.1.1 pointer_to_member_to_map_function_2args()	970
8.119.2 Member Function Documentation	970
8.119.2.1 operator()()	970
8.119.3 Member Data Documentation	970
8.119.3.1 ptr	970
8.119.3.2 c	970
8.119.3.3 arg1	971

8.119.3.4 arg2	971
8.120 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > Class Template Reference	971
8.120.1 Constructor & Destructor Documentation	973
8.120.1.1 pointer_to_member_to_map_function_3args()	973
8.120.2 Member Function Documentation	973
8.120.2.1 operator()()	973
8.120.3 Member Data Documentation	973
8.120.3.1 ptr	973
8.120.3.2 c	973
8.120.3.3 arg1	974
8.120.3.4 arg2	974
8.120.3.5 arg3	974
8.121 GiNaC::pole_error Class Reference	974
8.121.1 Detailed Description	975
8.121.2 Constructor & Destructor Documentation	975
8.121.2.1 pole_error()	975
8.121.3 Member Function Documentation	975
8.121.3.1 degree()	975
8.121.4 Member Data Documentation	976
8.121.4.1 deg	976
8.122 GiNaC::possymbol Class Reference	976
8.122.1 Detailed Description	981
8.122.2 Constructor & Destructor Documentation	981
8.122.2.1 possymbol() [1/3]	981
8.122.2.2 possymbol() [2/3]	981
8.122.2.3 possymbol() [3/3]	981
8.122.3 Member Function Documentation	982
8.122.3.1 get_domain()	982
8.122.3.2 duplicate()	982
8.123 GiNaC::power Class Reference	982
8.123.1 Detailed Description	987
8.123.2 Constructor & Destructor Documentation	987
8.123.2.1 power() [1/2]	987
8.123.2.2 power() [2/2]	987
8.123.3 Member Function Documentation	987
8.123.3.1 precedence()	987
8.123.3.2 info()	988
8.123.3.3 nops()	988
8.123.3.4 op()	988
8.123.3.5 map()	988
8.123.3.6 is_polynomial()	989
8.123.3.7 degree()	989

8.123.3.8 ldegree()	989
8.123.3.9 coeff()	989
8.123.3.10 eval()	990
8.123.3.11 evalf()	990
8.123.3.12 evalm()	991
8.123.3.13 series()	991
8.123.3.14 subs()	991
8.123.3.15 has()	992
8.123.3.16 normal()	992
8.123.3.17 to_rational()	992
8.123.3.18 to_polynomial()	993
8.123.3.19 conjugate()	993
8.123.3.20 real_part()	993
8.123.3.21 imag_part()	993
8.123.3.22 archive()	993
8.123.3.23 read_archive()	994
8.123.3.24 derivative()	994
8.123.3.25 eval_ncmul()	994
8.123.3.26 return_type()	994
8.123.3.27 return_type_tinfo()	995
8.123.3.28 expand()	995
8.123.3.29 print_power()	995
8.123.3.30 do_print_dflt()	995
8.123.3.31 do_print_latex()	996
8.123.3.32 do_print_csrc()	996
8.123.3.33 do_print_python()	996
8.123.3.34 do_print_python_repr()	996
8.123.3.35 do_print_csrc_cl_N()	996
8.123.3.36 expand_add()	997
8.123.3.37 expand_add_2()	997
8.123.3.38 expand_mul()	997
8.123.4 Friends And Related Symbol Documentation	998
8.123.4.1 mul	998
8.123.5 Member Data Documentation	998
8.123.5.1 basis	998
8.123.5.2 exponent	998
8.124 GiNaC::print_context Class Reference	999
8.124.1 Detailed Description	999
8.124.2 Constructor & Destructor Documentation	999
8.124.2.1 print_context()	999
8.124.2.2 ~print_context()	1000
8.124.3 Member Data Documentation	1000

8.124.3.1 s	1000
8.124.3.2 options	1000
8.125 GiNaC::print_context_options Class Reference	1001
8.125.1 Detailed Description	1001
8.125.2 Constructor & Destructor Documentation	1001
8.125.2.1 print_context_options()	1001
8.125.3 Member Function Documentation	1001
8.125.3.1 get_name()	1001
8.125.3.2 get_parent_name()	1002
8.125.3.3 get_id()	1002
8.125.4 Member Data Documentation	1002
8.125.4.1 name	1002
8.125.4.2 parent_name	1002
8.125.4.3 id	1002
8.126 GiNaC::print_csrc Class Reference	1003
8.126.1 Detailed Description	1004
8.126.2 Constructor & Destructor Documentation	1004
8.126.2.1 print_csrc()	1004
8.127 GiNaC::print_csrc_cl_N Class Reference	1004
8.127.1 Detailed Description	1005
8.127.2 Constructor & Destructor Documentation	1006
8.127.2.1 print_csrc_cl_N()	1006
8.128 GiNaC::print_csrc_double Class Reference	1006
8.128.1 Detailed Description	1007
8.128.2 Constructor & Destructor Documentation	1008
8.128.2.1 print_csrc_double()	1008
8.129 GiNaC::print_csrc_float Class Reference	1008
8.129.1 Detailed Description	1009
8.129.2 Constructor & Destructor Documentation	1010
8.129.2.1 print_csrc_float()	1010
8.130 GiNaC::print_dflt Class Reference	1010
8.130.1 Detailed Description	1011
8.130.2 Constructor & Destructor Documentation	1011
8.130.2.1 print_dflt()	1011
8.131 GiNaC::print_functor Class Reference	1011
8.131.1 Detailed Description	1012
8.131.2 Constructor & Destructor Documentation	1012
8.131.2.1 print_functor() [1/5]	1012
8.131.2.2 print_functor() [2/5]	1012
8.131.2.3 print_functor() [3/5]	1012
8.131.2.4 print_functor() [4/5]	1013
8.131.2.5 print_functor() [5/5]	1013

8.131.3 Member Function Documentation	1013
8.131.3.1 operator=()	1013
8.131.3.2 operator()()	1013
8.131.3.3 is_valid()	1013
8.131.4 Member Data Documentation	1013
8.131.4.1 impl	1013
8.132 GiNaC::print_functor_impl Class Reference	1014
8.132.1 Detailed Description	1014
8.132.2 Constructor & Destructor Documentation	1014
8.132.2.1 ~print_functor_impl()	1014
8.132.3 Member Function Documentation	1014
8.132.3.1 duplicate()	1014
8.132.3.2 operator()()	1015
8.133 GiNaC::print_latex Class Reference	1015
8.133.1 Detailed Description	1016
8.133.2 Constructor & Destructor Documentation	1016
8.133.2.1 print_latex()	1016
8.134 GiNaC::print_memfun_handler< T, C > Class Template Reference	1017
8.134.1 Detailed Description	1018
8.134.2 Member Typedef Documentation	1018
8.134.2.1 F	1018
8.134.3 Constructor & Destructor Documentation	1018
8.134.3.1 print_memfun_handler()	1018
8.134.4 Member Function Documentation	1018
8.134.4.1 duplicate()	1018
8.134.4.2 operator()()	1019
8.134.5 Member Data Documentation	1019
8.134.5.1 f	1019
8.135 GiNaC::print_options Class Reference	1019
8.135.1 Detailed Description	1019
8.135.2 Member Enumeration Documentation	1019
8.135.2.1 anonymous enum	1019
8.136 GiNaC::print_ptrfun_handler< T, C > Class Template Reference	1020
8.136.1 Detailed Description	1021
8.136.2 Member Typedef Documentation	1021
8.136.2.1 F	1021
8.136.3 Constructor & Destructor Documentation	1021
8.136.3.1 print_ptrfun_handler()	1021
8.136.4 Member Function Documentation	1021
8.136.4.1 duplicate()	1021
8.136.4.2 operator()()	1022
8.136.5 Member Data Documentation	1022

8.136.5.1 f . . . . .	1022
8.137 GiNaC::print_python Class Reference . . . . .	1022
8.137.1 Detailed Description . . . . .	1023
8.137.2 Constructor & Destructor Documentation . . . . .	1023
8.137.2.1 print_python() . . . . .	1023
8.138 GiNaC::print_python_repr Class Reference . . . . .	1024
8.138.1 Detailed Description . . . . .	1025
8.138.2 Constructor & Destructor Documentation . . . . .	1025
8.138.2.1 print_python_repr() . . . . .	1025
8.139 GiNaC::print_tree Class Reference . . . . .	1025
8.139.1 Detailed Description . . . . .	1026
8.139.2 Constructor & Destructor Documentation . . . . .	1026
8.139.2.1 print_tree() [1/2] . . . . .	1026
8.139.2.2 print_tree() [2/2] . . . . .	1027
8.139.3 Member Data Documentation . . . . .	1027
8.139.3.1 delta_indent . . . . .	1027
8.140 GiNaC::archive_node::property Struct Reference . . . . .	1027
8.140.1 Detailed Description . . . . .	1028
8.140.2 Constructor & Destructor Documentation . . . . .	1028
8.140.2.1 property() [1/2] . . . . .	1028
8.140.2.2 property() [2/2] . . . . .	1028
8.140.3 Member Data Documentation . . . . .	1028
8.140.3.1 type . . . . .	1028
8.140.3.2 name . . . . .	1028
8.140.3.3 value . . . . .	1028
8.141 GiNaC::archive_node::property_info Struct Reference . . . . .	1029
8.141.1 Detailed Description . . . . .	1029
8.141.2 Constructor & Destructor Documentation . . . . .	1029
8.141.2.1 property_info() [1/2] . . . . .	1029
8.141.2.2 property_info() [2/2] . . . . .	1029
8.141.3 Member Data Documentation . . . . .	1030
8.141.3.1 type . . . . .	1030
8.141.3.2 name . . . . .	1030
8.141.3.3 count . . . . .	1030
8.142 GiNaC::pseries Class Reference . . . . .	1030
8.142.1 Detailed Description . . . . .	1035
8.142.2 Constructor & Destructor Documentation . . . . .	1035
8.142.2.1 pseries() [1/2] . . . . .	1035
8.142.2.2 pseries() [2/2] . . . . .	1036
8.142.3 Member Function Documentation . . . . .	1036
8.142.3.1 precedence() . . . . .	1036
8.142.3.2 nops() . . . . .	1036

8.142.3.3 op()	1036
8.142.3.4 degree()	1037
8.142.3.5 ldegree()	1037
8.142.3.6 coeff()	1037
8.142.3.7 collect()	1038
8.142.3.8 eval()	1038
8.142.3.9 evalf()	1038
8.142.3.10 series()	1038
8.142.3.11 subs()	1039
8.142.3.12 normal()	1039
8.142.3.13 expand()	1039
8.142.3.14 conjugate()	1040
8.142.3.15 real_part()	1040
8.142.3.16 imag_part()	1040
8.142.3.17 eval_integ()	1040
8.142.3.18 evalm()	1040
8.142.3.19 archive()	1041
8.142.3.20 read_archive()	1041
8.142.3.21 derivative()	1041
8.142.3.22 get_var()	1041
8.142.3.23 get_point()	1042
8.142.3.24 convert_to_poly()	1042
8.142.3.25 is_compatible_to()	1042
8.142.3.26 is_zero()	1042
8.142.3.27 is_terminating()	1042
8.142.3.28 coeffop()	1043
8.142.3.29 exponop()	1043
8.142.3.30 add_series()	1043
8.142.3.31 mul_const()	1043
8.142.3.32 mul_series()	1044
8.142.3.33 power_const()	1044
8.142.3.34 shift_exponents()	1044
8.142.3.35 print_series()	1045
8.142.3.36 do_print()	1045
8.142.3.37 do_print_latex()	1045
8.142.3.38 do_print_tree()	1045
8.142.3.39 do_print_python()	1046
8.142.3.40 do_print_python_repr()	1046
8.142.4 Member Data Documentation	1046
8.142.4.1 seq	1046
8.142.4.2 var	1046
8.142.4.3 point	1047



8.143 GiNaC::psi1_SERIAL Class Reference	1047
8.143.1 Detailed Description	1047
8.143.2 Member Data Documentation	1047
8.143.2.1 serial	1047
8.144 GiNaC::psi2_SERIAL Class Reference	1048
8.144.1 Detailed Description	1048
8.144.2 Member Data Documentation	1048
8.144.2.1 serial	1048
8.145 GiNaC::ptr< T > Class Template Reference	1048
8.145.1 Detailed Description	1049
8.145.2 Constructor & Destructor Documentation	1050
8.145.2.1 ptr() [1/3]	1050
8.145.2.2 ptr() [2/3]	1050
8.145.2.3 ptr() [3/3]	1050
8.145.2.4 ~ptr()	1050
8.145.3 Member Function Documentation	1050
8.145.3.1 operator=()	1050
8.145.3.2 operator*()	1051
8.145.3.3 operator->()	1051
8.145.3.4 makewritable()	1051
8.145.3.5 swap()	1051
8.145.3.6 operator==(())	1051
8.145.3.7 operator"!=(())	1051
8.145.4 Friends And Related Symbol Documentation	1052
8.145.4.1 std::less< ptr< T > >	1052
8.145.4.2 get_pointer	1052
8.145.4.3 operator== [1/2]	1052
8.145.4.4 operator"!=[1/2]	1052
8.145.4.5 operator== [2/2]	1052
8.145.4.6 operator"!=[2/2]	1053
8.145.4.7 operator<<	1053
8.145.5 Member Data Documentation	1053
8.145.5.1 p	1053
8.146 GiNaC::realsymbol Class Reference	1053
8.146.1 Detailed Description	1058
8.146.2 Constructor & Destructor Documentation	1058
8.146.2.1 realsymbol() [1/3]	1058
8.146.2.2 realsymbol() [2/3]	1059
8.146.2.3 realsymbol() [3/3]	1059
8.146.3 Member Function Documentation	1059
8.146.3.1 get_domain()	1059
8.146.3.2 conjugate()	1059

8.146.3.3	<a href="#">real_part()</a>	1059
8.146.3.4	<a href="#">imag_part()</a>	1059
8.146.3.5	<a href="#">duplicate()</a>	1060
8.147	<a href="#">GiNaC::refcounted Class Reference</a>	1060
8.147.1	<a href="#">Detailed Description</a>	1062
8.147.2	<a href="#">Constructor &amp; Destructor Documentation</a>	1062
8.147.2.1	<a href="#">refcounted()</a>	1062
8.147.3	<a href="#">Member Function Documentation</a>	1062
8.147.3.1	<a href="#">add_reference()</a>	1062
8.147.3.2	<a href="#">remove_reference()</a>	1062
8.147.3.3	<a href="#">get_refcount()</a>	1062
8.147.3.4	<a href="#">set_refcount()</a>	1062
8.147.4	<a href="#">Member Data Documentation</a>	1063
8.147.4.1	<a href="#">refcount</a>	1063
8.148	<a href="#">GiNaC::registered_class_options Class Reference</a>	1063
8.148.1	<a href="#">Detailed Description</a>	1063
8.148.2	<a href="#">Constructor &amp; Destructor Documentation</a>	1064
8.148.2.1	<a href="#">registered_class_options()</a>	1064
8.148.3	<a href="#">Member Function Documentation</a>	1064
8.148.3.1	<a href="#">get_name()</a>	1064
8.148.3.2	<a href="#">get_parent_name()</a>	1064
8.148.3.3	<a href="#">get_id()</a>	1064
8.148.3.4	<a href="#">get_print_dispatch_table()</a>	1064
8.148.3.5	<a href="#">print_func()</a> [1/3]	1064
8.148.3.6	<a href="#">print_func()</a> [2/3]	1065
8.148.3.7	<a href="#">print_func()</a> [3/3]	1065
8.148.3.8	<a href="#">set_print_func()</a>	1065
8.148.4	<a href="#">Member Data Documentation</a>	1065
8.148.4.1	<a href="#">name</a>	1065
8.148.4.2	<a href="#">parent_name</a>	1065
8.148.4.3	<a href="#">tinfo_key</a>	1065
8.148.4.4	<a href="#">print_dispatch_table</a>	1066
8.149	<a href="#">GiNaC::relational Class Reference</a>	1066
8.149.1	<a href="#">Detailed Description</a>	1071
8.149.2	<a href="#">Member Typedef Documentation</a>	1071
8.149.2.1	<a href="#">safe_bool</a>	1071
8.149.3	<a href="#">Member Enumeration Documentation</a>	1071
8.149.3.1	<a href="#">operators</a>	1071
8.149.4	<a href="#">Constructor &amp; Destructor Documentation</a>	1072
8.149.4.1	<a href="#">relational()</a>	1072
8.149.5	<a href="#">Member Function Documentation</a>	1072
8.149.5.1	<a href="#">precedence()</a>	1072

8.149.5.2 info()	1072
8.149.5.3 nops()	1073
8.149.5.4 op()	1073
8.149.5.5 map()	1073
8.149.5.6 subs()	1073
8.149.5.7 archive()	1073
8.149.5.8 read_archive()	1074
8.149.5.9 canonical()	1074
8.149.5.10 eval_ncmul()	1074
8.149.5.11 match_same_type()	1074
8.149.5.12 return_type()	1075
8.149.5.13 return_type_tinfo()	1075
8.149.5.14 calchash()	1075
8.149.5.15 do_print()	1075
8.149.5.16 do_print_python_repr()	1075
8.149.5.17 lhs()	1076
8.149.5.18 rhs()	1076
8.149.5.19 make_safe_bool()	1076
8.149.5.20 operator safe_bool()	1076
8.149.5.21 operator"!()	1076
8.149.6 Member Data Documentation	1077
8.149.6.1 lh	1077
8.149.6.2 rh	1077
8.149.6.3 o	1077
8.150 GiNaC::remember_strategies Class Reference	1077
8.150.1 Detailed Description	1077
8.150.2 Member Enumeration Documentation	1077
8.150.2.1 anonymous enum	1077
8.151 GiNaC::remember_table Class Reference	1078
8.151.1 Detailed Description	1079
8.151.2 Constructor & Destructor Documentation	1079
8.151.2.1 remember_table() [1/2]	1079
8.151.2.2 remember_table() [2/2]	1080
8.151.3 Member Function Documentation	1080
8.151.3.1 lookup_entry()	1080
8.151.3.2 add_entry()	1080
8.151.3.3 clear_all_entries()	1080
8.151.3.4 show_statistics()	1080
8.151.3.5 remember_tables()	1080
8.151.3.6 init_table()	1081
8.151.4 Member Data Documentation	1081
8.151.4.1 table_size	1081

8.151.4.2 max_assoc_size . . . . .	1081
8.151.4.3 remember_strategy . . . . .	1081
8.152 GiNaC::remember_table_entry Class Reference . . . . .	1082
8.152.1 Detailed Description . . . . .	1083
8.152.2 Constructor & Destructor Documentation . . . . .	1083
8.152.2.1 remember_table_entry() . . . . .	1083
8.152.3 Member Function Documentation . . . . .	1083
8.152.3.1 is_equal() . . . . .	1083
8.152.3.2 get_result() . . . . .	1083
8.152.3.3 get_last_access() . . . . .	1083
8.152.3.4 get_successful_hits() . . . . .	1084
8.152.4 Member Data Documentation . . . . .	1084
8.152.4.1 hashvalue . . . . .	1084
8.152.4.2 seq . . . . .	1084
8.152.4.3 result . . . . .	1084
8.152.4.4 last_access . . . . .	1084
8.152.4.5 successful_hits . . . . .	1084
8.152.4.6 access_counter . . . . .	1084
8.153 GiNaC::remember_table_list Class Reference . . . . .	1085
8.153.1 Detailed Description . . . . .	1086
8.153.2 Constructor & Destructor Documentation . . . . .	1086
8.153.2.1 remember_table_list() . . . . .	1086
8.153.3 Member Function Documentation . . . . .	1086
8.153.3.1 add_entry() . . . . .	1086
8.153.3.2 lookup_entry() . . . . .	1086
8.153.4 Member Data Documentation . . . . .	1086
8.153.4.1 max_assoc_size . . . . .	1086
8.153.4.2 remember_strategy . . . . .	1086
8.154 GiNaC::return_type_t Struct Reference . . . . .	1087
8.154.1 Detailed Description . . . . .	1087
8.154.2 Member Function Documentation . . . . .	1087
8.154.2.1 operator<() . . . . .	1087
8.154.2.2 operator==( ) . . . . .	1087
8.154.2.3 operator"!=( ) . . . . .	1088
8.154.3 Member Data Documentation . . . . .	1088
8.154.3.1 tinfo . . . . .	1088
8.154.3.2 rl . . . . .	1088
8.155 GiNaC::return_types Class Reference . . . . .	1088
8.155.1 Member Enumeration Documentation . . . . .	1088
8.155.1.1 anonymous enum . . . . .	1088
8.156 GiNaC::relational::safe_bool_helper Struct Reference . . . . .	1089
8.156.1 Member Function Documentation . . . . .	1089

8.156.1.1 nonnull()	1089
8.157 GiNaC::scalar_products Class Reference	1089
8.157.1 Detailed Description	1090
8.157.2 Member Function Documentation	1090
8.157.2.1 add() [1/2]	1090
8.157.2.2 add() [2/2]	1090
8.157.2.3 add_vectors()	1090
8.157.2.4 clear()	1091
8.157.2.5 is_defined()	1091
8.157.2.6 evaluate()	1091
8.157.2.7 debugprint()	1091
8.157.3 Member Data Documentation	1091
8.157.3.1 spm	1091
8.158 GiNaC::series_options Class Reference	1092
8.158.1 Detailed Description	1092
8.158.2 Member Enumeration Documentation	1092
8.158.2.1 anonymous enum	1092
8.159 GiNaC::solve_algo Class Reference	1092
8.159.1 Detailed Description	1092
8.159.2 Member Enumeration Documentation	1092
8.159.2.1 anonymous enum	1092
8.160 GiNaC::spinidx Class Reference	1093
8.160.1 Detailed Description	1100
8.160.2 Constructor & Destructor Documentation	1100
8.160.2.1 spinidx()	1100
8.160.3 Member Function Documentation	1101
8.160.3.1 is_dummy_pair_same_type()	1101
8.160.3.2 conjugate()	1101
8.160.3.3 archive()	1101
8.160.3.4 read_archive()	1102
8.160.3.5 match_same_type()	1102
8.160.3.6 is_dotted()	1102
8.160.3.7 is_undotted()	1103
8.160.3.8 toggle_dot()	1103
8.160.3.9 toggle_variance_dot()	1103
8.160.3.10 do_print()	1103
8.160.3.11 do_print_latex()	1103
8.160.3.12 do_print_tree()	1103
8.160.4 Member Data Documentation	1104
8.160.4.1 dotted	1104
8.161 GiNaC::spinmetric Class Reference	1104
8.161.1 Detailed Description	1109

8.161.2 Member Function Documentation	1109
8.161.2.1 info()	1109
8.161.2.2 eval_indexed()	1109
8.161.2.3 contract_with()	1110
8.161.2.4 do_print()	1110
8.161.2.5 do_print_latex()	1110
8.162 GiNaC::spmapkey Class Reference	1110
8.162.1 Constructor & Destructor Documentation	1112
8.162.1.1 spmapkey() [1/2]	1112
8.162.1.2 spmapkey() [2/2]	1112
8.162.2 Member Function Documentation	1112
8.162.2.1 operator==(())	1112
8.162.2.2 operator<()	1112
8.162.2.3 debugprint()	1112
8.162.3 Member Data Documentation	1112
8.162.3.1 v1	1112
8.162.3.2 v2	1113
8.162.3.3 dim	1113
8.163 GiNaC::status_flags Class Reference	1113
8.163.1 Detailed Description	1113
8.163.2 Member Enumeration Documentation	1113
8.163.2.1 anonymous enum	1113
8.164 GiNaC::structure< T, ComparisonPolicy > Class Template Reference	1114
8.164.1 Detailed Description	1118
8.164.2 Constructor & Destructor Documentation	1119
8.164.2.1 structure()	1119
8.164.3 Member Function Documentation	1119
8.164.3.1 get_class_name()	1119
8.164.3.2 eval()	1119
8.164.3.3 evalm()	1119
8.164.3.4 eval_ncmul()	1120
8.164.3.5 eval_indexed()	1120
8.164.3.6 print()	1120
8.164.3.7 precedence()	1120
8.164.3.8 info()	1121
8.164.3.9 nops()	1121
8.164.3.10 op()	1121
8.164.3.11 operator[]() [1/4]	1121
8.164.3.12 operator[]() [2/4]	1121
8.164.3.13 let_op()	1122
8.164.3.14 operator[]() [3/4]	1122
8.164.3.15 operator[]() [4/4]	1122

8.164.3.16	has()	1122
8.164.3.17	match()	1123
8.164.3.18	match_same_type()	1123
8.164.3.19	subs()	1123
8.164.3.20	map()	1124
8.164.3.21	degree()	1124
8.164.3.22	ldegree()	1124
8.164.3.23	coeff()	1124
8.164.3.24	expand()	1125
8.164.3.25	collect()	1125
8.164.3.26	derivative()	1125
8.164.3.27	series()	1126
8.164.3.28	normal()	1126
8.164.3.29	to_rational()	1126
8.164.3.30	to_polynomial()	1127
8.164.3.31	integer_content()	1127
8.164.3.32	smod()	1127
8.164.3.33	max_coefficient()	1128
8.164.3.34	get_free_indices()	1128
8.164.3.35	add_indexed()	1128
8.164.3.36	scalar_mul_indexed()	1129
8.164.3.37	contract_with()	1130
8.164.3.38	return_type()	1131
8.164.3.39	return_type_tinfo()	1131
8.164.3.40	is_equal_same_type()	1131
8.164.3.41	calchash()	1131
8.164.3.42	operator->()	1132
8.164.3.43	get_struct() [1/2]	1132
8.164.3.44	get_struct() [2/2]	1132
8.164.4	Member Data Documentation	1132
8.164.4.1	obj	1132
8.165	GiNaC::su3d Class Reference	1133
8.165.1	Detailed Description	1137
8.165.2	Member Function Documentation	1137
8.165.2.1	eval_indexed()	1137
8.165.2.2	contract_with()	1137
8.165.2.3	return_type()	1138
8.165.2.4	do_print()	1138
8.165.2.5	do_print_latex()	1138
8.166	GiNaC::su3f Class Reference	1138
8.166.1	Detailed Description	1142
8.166.2	Member Function Documentation	1143

8.166.2.1 eval_indexed()	1143
8.166.2.2 contract_with()	1143
8.166.2.3 return_type()	1143
8.166.2.4 do_print()	1143
8.166.2.5 do_print_latex()	1143
8.167 GiNaC::su3one Class Reference	1144
8.167.1 Detailed Description	1148
8.167.2 Member Function Documentation	1148
8.167.2.1 do_print()	1148
8.167.2.2 do_print_latex()	1148
8.168 GiNaC::su3t Class Reference	1149
8.168.1 Detailed Description	1153
8.168.2 Member Function Documentation	1153
8.168.2.1 contract_with()	1153
8.168.2.2 do_print()	1153
8.168.2.3 do_print_latex()	1154
8.169 GiNaC::subs_options Class Reference	1154
8.169.1 Detailed Description	1154
8.169.2 Member Enumeration Documentation	1154
8.169.2.1 anonymous enum	1154
8.170 GiNaC::sy_is_less Class Reference	1155
8.170.1 Constructor & Destructor Documentation	1155
8.170.1.1 sy_is_less()	1155
8.170.2 Member Function Documentation	1155
8.170.2.1 operator()()	1155
8.170.3 Member Data Documentation	1155
8.170.3.1 v	1155
8.171 GiNaC::sy_swap Class Reference	1155
8.171.1 Constructor & Destructor Documentation	1156
8.171.1.1 sy_swap()	1156
8.171.2 Member Function Documentation	1156
8.171.2.1 operator()()	1156
8.171.3 Member Data Documentation	1156
8.171.3.1 v	1156
8.171.3.2 swapped	1156
8.172 GiNaC::sym_desc Struct Reference	1157
8.172.1 Detailed Description	1158
8.172.2 Constructor & Destructor Documentation	1158
8.172.2.1 sym_desc()	1158
8.172.3 Member Function Documentation	1158
8.172.3.1 operator<()	1158
8.172.4 Member Data Documentation	1159



8.172.4.1 sym	1159
8.172.4.2 deg_a	1159
8.172.4.3 deg_b	1159
8.172.4.4 ldeg_a	1159
8.172.4.5 ldeg_b	1159
8.172.4.6 max_deg	1159
8.172.4.7 max_lcnops	1160
8.173 GiNaC::symbol Class Reference	1160
8.173.1 Detailed Description	1164
8.173.2 Constructor & Destructor Documentation	1165
8.173.2.1 symbol() [1/2]	1165
8.173.2.2 symbol() [2/2]	1165
8.173.3 Member Function Documentation	1165
8.173.3.1 info()	1165
8.173.3.2 eval()	1165
8.173.3.3 evalf()	1166
8.173.3.4 series()	1166
8.173.3.5 subs()	1166
8.173.3.6 normal()	1166
8.173.3.7 to_rational()	1167
8.173.3.8 to_polynomial()	1167
8.173.3.9 conjugate()	1167
8.173.3.10 real_part()	1167
8.173.3.11 imag_part()	1167
8.173.3.12 is_polynomial()	1168
8.173.3.13 archive()	1168
8.173.3.14 read_archive()	1168
8.173.3.15 derivative()	1168
8.173.3.16 is_equal_same_type()	1169
8.173.3.17 calchash()	1169
8.173.3.18 get_domain()	1169
8.173.3.19 set_name()	1169
8.173.3.20 set_TeX_name()	1170
8.173.3.21 get_name()	1170
8.173.3.22 get_TeX_name()	1170
8.173.3.23 do_print()	1170
8.173.3.24 do_print_latex()	1170
8.173.3.25 do_print_tree()	1170
8.173.3.26 do_print_python_repr()	1171
8.173.4 Member Data Documentation	1171
8.173.4.1 serial	1171
8.173.4.2 name	1171

8.173.4.3 TeX_name	1171
8.173.4.4 next_serial	1171
8.174 GiNaC::symbolset Class Reference	1172
8.174.1 Constructor & Destructor Documentation	1172
8.174.1.1 symbolset()	1172
8.174.2 Member Function Documentation	1172
8.174.2.1 insert_symbols()	1172
8.174.2.2 has()	1172
8.174.3 Member Data Documentation	1173
8.174.3.1 s	1173
8.175 GiNaC::symmetry Class Reference	1173
8.175.1 Detailed Description	1178
8.175.2 Member Enumeration Documentation	1178
8.175.2.1 symmetry_type	1178
8.175.3 Constructor & Destructor Documentation	1178
8.175.3.1 symmetry() [1/2]	1178
8.175.3.2 symmetry() [2/2]	1179
8.175.4 Member Function Documentation	1179
8.175.4.1 archive()	1179
8.175.4.2 read_archive()	1179
8.175.4.3 calchash()	1180
8.175.4.4 get_type()	1180
8.175.4.5 set_type()	1180
8.175.4.6 add()	1180
8.175.4.7 validate()	1181
8.175.4.8 has_symmetry()	1181
8.175.4.9 has_nonsymmetric()	1181
8.175.4.10 has_cyclic()	1181
8.175.4.11 do_print()	1181
8.175.4.12 do_print_tree()	1182
8.175.5 Friends And Related Symbol Documentation	1182
8.175.5.1 sy_is_less	1182
8.175.5.2 sy_swap	1182
8.175.5.3 canonicalize	1182
8.175.6 Member Data Documentation	1183
8.175.6.1 type	1183
8.175.6.2 indices	1183
8.175.6.3 children	1183
8.176 GiNaC::symminfo Class Reference	1183
8.176.1 Detailed Description	1185
8.176.2 Constructor & Destructor Documentation	1185
8.176.2.1 symminfo() [1/2]	1185

8.176.2.2 symminfo() [2/2]	1185
8.176.3 Member Data Documentation	1185
8.176.3.1 symmterm	1185
8.176.3.2 coeff	1185
8.176.3.3 orig	1185
8.176.3.4 num	1186
8.177 GiNaC::symminfo_is_less_by_orig Class Reference	1186
8.177.1 Member Function Documentation	1186
8.177.1.1 operator()	1186
8.178 GiNaC::symminfo_is_less_by_symmterm Class Reference	1186
8.178.1 Member Function Documentation	1186
8.178.1.1 operator()	1186
8.179 GiNaC::tensdelta Class Reference	1187
8.179.1 Detailed Description	1191
8.179.2 Member Function Documentation	1191
8.179.2.1 info()	1191
8.179.2.2 eval_indexed()	1192
8.179.2.3 contract_with()	1192
8.179.2.4 return_type()	1192
8.179.2.5 do_print()	1192
8.179.2.6 do_print_latex()	1192
8.180 GiNaC::tensepsilon Class Reference	1193
8.180.1 Detailed Description	1197
8.180.2 Constructor & Destructor Documentation	1197
8.180.2.1 tensepsilon()	1197
8.180.3 Member Function Documentation	1198
8.180.3.1 info()	1198
8.180.3.2 eval_indexed()	1198
8.180.3.3 contract_with()	1198
8.180.3.4 archive()	1198
8.180.3.5 read_archive()	1199
8.180.3.6 return_type()	1199
8.180.3.7 do_print()	1199
8.180.3.8 do_print_latex()	1199
8.180.4 Member Data Documentation	1199
8.180.4.1 minkowski	1199
8.180.4.2 pos_sig	1200
8.181 GiNaC::tensmetric Class Reference	1200
8.181.1 Detailed Description	1204
8.181.2 Member Function Documentation	1205
8.181.2.1 info()	1205
8.181.2.2 eval_indexed()	1205

8.181.2.3 <code>contract_with()</code> . . . . .	1205
8.181.2.4 <code>return_type()</code> . . . . .	1205
8.181.2.5 <code>do_print()</code> . . . . .	1206
8.182 <code>GiNaC::tensor</code> Class Reference . . . . .	1206
8.182.1 Detailed Description . . . . .	1210
8.182.2 Member Function Documentation . . . . .	1210
8.182.2.1 <code>return_type()</code> . . . . .	1210
8.182.2.2 <code>replace_contr_index()</code> . . . . .	1211
8.183 <code>GiNaC::terminfo</code> Class Reference . . . . .	1211
8.183.1 Detailed Description . . . . .	1212
8.183.2 Constructor & Destructor Documentation . . . . .	1212
8.183.2.1 <code>terminfo()</code> . . . . .	1212
8.183.3 Member Data Documentation . . . . .	1212
8.183.3.1 <code>orig</code> . . . . .	1212
8.183.3.2 <code>symm</code> . . . . .	1212
8.184 <code>GiNaC::terminfo_is_less</code> Class Reference . . . . .	1213
8.184.1 Member Function Documentation . . . . .	1213
8.184.1.1 <code>operator()()</code> . . . . .	1213
8.185 <code>GiNaC::Gt_detail::TransformExpressionWithCache&lt; Type &gt;</code> Class Template Reference . . . . .	1213
8.185.1 Constructor & Destructor Documentation . . . . .	1214
8.185.1.1 <code>TransformExpressionWithCache()</code> . . . . .	1214
8.185.2 Member Function Documentation . . . . .	1214
8.185.2.1 <code>operator()()</code> . . . . .	1214
8.185.2.2 <code>print_cache()</code> . . . . .	1215
8.185.2.3 <code>get_cache()</code> . . . . .	1215
8.185.2.4 <code>clear_cache()</code> . . . . .	1215
8.185.2.5 <code>check_type()</code> [1/2] . . . . .	1215
8.185.2.6 <code>check_type()</code> [2/2] . . . . .	1215
8.185.2.7 <code>impl()</code> . . . . .	1215
8.185.3 Member Data Documentation . . . . .	1216
8.185.3.1 <code>func_obj</code> . . . . .	1216
8.185.3.2 <code>func_default</code> . . . . .	1216
8.185.3.3 <code>cache</code> . . . . .	1216
8.186 <code>GiNaC::class_info&lt; OPT &gt;::tree_node</code> Struct Reference . . . . .	1216
8.186.1 Constructor & Destructor Documentation . . . . .	1217
8.186.1.1 <code>tree_node()</code> . . . . .	1217
8.186.2 Member Function Documentation . . . . .	1217
8.186.2.1 <code>add_child()</code> . . . . .	1217
8.186.3 Member Data Documentation . . . . .	1217
8.186.3.1 <code>children</code> . . . . .	1217
8.186.3.2 <code>info</code> . . . . .	1217
8.187 <code>GiNaC::unarchive_table_t</code> Class Reference . . . . .	1218

8.187.1 Constructor & Destructor Documentation	1218
8.187.1.1 unarchive_table_t()	1218
8.187.1.2 ~unarchive_table_t()	1218
8.187.2 Member Function Documentation	1218
8.187.2.1 find()	1218
8.187.2.2 insert()	1218
8.187.3 Member Data Documentation	1219
8.187.3.1 usecount	1219
8.187.3.2 unarch_map	1219
8.188 GiNaC::user_defined_kernel Class Reference	1219
8.188.1 Detailed Description	1224
8.188.2 Constructor & Destructor Documentation	1225
8.188.2.1 user_defined_kernel()	1225
8.188.3 Member Function Documentation	1225
8.188.3.1 nops()	1225
8.188.3.2 op()	1225
8.188.3.3 let_op()	1225
8.188.3.4 is_numeric()	1225
8.188.3.5 Laurent_series()	1226
8.188.3.6 uses_Laurent_series()	1226
8.188.3.7 do_print()	1226
8.188.4 Member Data Documentation	1226
8.188.4.1 f	1226
8.188.4.2 x	1226
8.189 GiNaC::varidx Class Reference	1227
8.189.1 Detailed Description	1233
8.189.2 Constructor & Destructor Documentation	1233
8.189.2.1 varidx()	1233
8.189.3 Member Function Documentation	1233
8.189.3.1 is_dummy_pair_same_type()	1233
8.189.3.2 archive()	1233
8.189.3.3 read_archive()	1234
8.189.3.4 match_same_type()	1234
8.189.3.5 is_covariant()	1234
8.189.3.6 is_contravariant()	1235
8.189.3.7 toggle_variance()	1235
8.189.3.8 do_print()	1235
8.189.3.9 do_print_tree()	1235
8.189.4 Member Data Documentation	1235
8.189.4.1 covariant	1235
8.190 GiNaC::visitor Class Reference	1236
8.190.1 Detailed Description	1236

8.190.2 Constructor & Destructor Documentation	1236
8.190.2.1 ~visitor()	1236
8.191 GiNaC::wildcard Class Reference	1236
8.191.1 Detailed Description	1240
8.191.2 Constructor & Destructor Documentation	1240
8.191.2.1 wildcard()	1240
8.191.3 Member Function Documentation	1241
8.191.3.1 match()	1241
8.191.3.2 archive()	1241
8.191.3.3 read_archive()	1241
8.191.3.4 calchash()	1241
8.191.3.5 get_label()	1242
8.191.3.6 do_print()	1242
8.191.3.7 do_print_tree()	1242
8.191.3.8 do_print_python_repr()	1242
8.191.4 Member Data Documentation	1242
8.191.4.1 label	1242
8.192 GiNaC::zeta1_SERIAL Class Reference	1243
8.192.1 Detailed Description	1243
8.192.2 Member Data Documentation	1243
8.192.2.1 serial	1243
8.193 GiNaC::zeta2_SERIAL Class Reference	1243
8.193.1 Detailed Description	1244
8.193.2 Member Data Documentation	1244
8.193.2.1 serial	1244
<b>9 File Documentation</b>	<b>1245</b>
9.1 add.cpp File Reference	1245
9.1.1 Detailed Description	1245
9.2 add.h File Reference	1246
9.2.1 Detailed Description	1246
9.3 archive.cpp File Reference	1246
9.3.1 Detailed Description	1247
9.4 archive.h File Reference	1247
9.4.1 Detailed Description	1248
9.4.2 Macro Definition Documentation	1248
9.4.2.1 GINAC_DECLARE_UNARCHIVER	1248
9.4.2.2 GINAC_BIND_UNARCHIVER	1249
9.5 assertion.h File Reference	1249
9.5.1 Detailed Description	1250
9.5.2 Macro Definition Documentation	1250
9.5.2.1 GINAC_ASSERT	1250

9.6 basic.cpp File Reference . . . . .	1251
9.6.1 Detailed Description . . . . .	1252
9.7 basic.h File Reference . . . . .	1252
9.7.1 Detailed Description . . . . .	1253
9.8 class_info.h File Reference . . . . .	1253
9.8.1 Detailed Description . . . . .	1254
9.9 clifford.cpp File Reference . . . . .	1254
9.9.1 Detailed Description . . . . .	1256
9.10 clifford.h File Reference . . . . .	1256
9.10.1 Detailed Description . . . . .	1258
9.11 color.cpp File Reference . . . . .	1258
9.11.1 Detailed Description . . . . .	1259
9.11.2 Macro Definition Documentation . . . . .	1259
9.11.2.1 TEST_PERMUTATION . . . . .	1259
9.11.2.2 CMPINDICES . . . . .	1260
9.12 color.h File Reference . . . . .	1260
9.12.1 Detailed Description . . . . .	1261
9.13 compiler.h File Reference . . . . .	1261
9.13.1 Detailed Description . . . . .	1261
9.13.2 Macro Definition Documentation . . . . .	1262
9.13.2.1 unlikely . . . . .	1262
9.13.2.2 likely . . . . .	1262
9.13.2.3 attribute_pure . . . . .	1262
9.14 constant.cpp File Reference . . . . .	1262
9.14.1 Detailed Description . . . . .	1263
9.15 constant.h File Reference . . . . .	1263
9.15.1 Detailed Description . . . . .	1263
9.16 container.h File Reference . . . . .	1264
9.16.1 Detailed Description . . . . .	1264
9.17 crc32.h File Reference . . . . .	1264
9.17.1 Detailed Description . . . . .	1265
9.18 ex.cpp File Reference . . . . .	1265
9.18.1 Detailed Description . . . . .	1265
9.19 ex.h File Reference . . . . .	1265
9.19.1 Detailed Description . . . . .	1267
9.20 excompiler.cpp File Reference . . . . .	1268
9.20.1 Detailed Description . . . . .	1268
9.21 excompiler.h File Reference . . . . .	1268
9.21.1 Detailed Description . . . . .	1269
9.22 expair.cpp File Reference . . . . .	1269
9.22.1 Detailed Description . . . . .	1270
9.23 expair.h File Reference . . . . .	1270

9.23.1 Detailed Description	1270
9.24 expairseq.cpp File Reference	1271
9.24.1 Detailed Description	1271
9.25 expairseq.h File Reference	1271
9.25.1 Detailed Description	1272
9.26 exprseq.cpp File Reference	1272
9.26.1 Detailed Description	1272
9.27 exprseq.h File Reference	1273
9.27.1 Detailed Description	1273
9.28 factor.cpp File Reference	1273
9.28.1 Detailed Description	1274
9.28.2 Macro Definition Documentation	1274
9.28.2.1 DCOUT	1274
9.28.2.2 DCOUTVAR	1274
9.28.2.3 DCOUT2	1274
9.28.2.4 USE_SAME_DEGREE_FACTOR	1275
9.29 factor.h File Reference	1275
9.29.1 Detailed Description	1275
9.30 fail.cpp File Reference	1275
9.30.1 Detailed Description	1275
9.31 fail.h File Reference	1276
9.31.1 Detailed Description	1276
9.32 fderivative.cpp File Reference	1276
9.32.1 Detailed Description	1277
9.33 fderivative.h File Reference	1277
9.33.1 Detailed Description	1277
9.34 flags.h File Reference	1277
9.34.1 Detailed Description	1278
9.35 function.cpp File Reference	1278
9.35.1 Detailed Description	1279
9.36 function.h File Reference	1279
9.36.1 Detailed Description	1286
9.36.2 Macro Definition Documentation	1286
9.36.2.1 DECLARE_FUNCTION_1P	1286
9.36.2.2 DECLARE_FUNCTION_2P	1286
9.36.2.3 DECLARE_FUNCTION_3P	1286
9.36.2.4 DECLARE_FUNCTION_4P	1287
9.36.2.5 DECLARE_FUNCTION_5P	1287
9.36.2.6 DECLARE_FUNCTION_6P	1287
9.36.2.7 DECLARE_FUNCTION_7P	1287
9.36.2.8 DECLARE_FUNCTION_8P	1288
9.36.2.9 DECLARE_FUNCTION_9P	1288



9.36.2.10 DECLARE_FUNCTION_10P . . . . .	1288
9.36.2.11 DECLARE_FUNCTION_11P . . . . .	1288
9.36.2.12 DECLARE_FUNCTION_12P . . . . .	1289
9.36.2.13 DECLARE_FUNCTION_13P . . . . .	1289
9.36.2.14 DECLARE_FUNCTION_14P . . . . .	1289
9.36.2.15 REGISTER_FUNCTION . . . . .	1289
9.36.2.16 is_ex_the_function . . . . .	1290
9.37 ginac.h File Reference . . . . .	1290
9.37.1 Detailed Description . . . . .	1291
9.38 Gt.cpp File Reference . . . . .	1291
9.38.1 Detailed Description . . . . .	1291
9.38.2 Macro Definition Documentation . . . . .	1292
9.38.2.1 DEBUG_GT . . . . .	1292
9.39 Gt.h File Reference . . . . .	1292
9.39.1 Detailed Description . . . . .	1292
9.40 Gt_helpers.cpp File Reference . . . . .	1292
9.40.1 Detailed Description . . . . .	1293
9.41 Gt_helpers.h File Reference . . . . .	1293
9.41.1 Detailed Description . . . . .	1294
9.42 hash_map.h File Reference . . . . .	1294
9.42.1 Detailed Description . . . . .	1294
9.43 hash_seed.h File Reference . . . . .	1294
9.43.1 Detailed Description . . . . .	1295
9.44 idx.cpp File Reference . . . . .	1295
9.44.1 Detailed Description . . . . .	1296
9.45 idx.h File Reference . . . . .	1296
9.45.1 Detailed Description . . . . .	1297
9.46 indexed.cpp File Reference . . . . .	1297
9.46.1 Detailed Description . . . . .	1298
9.47 indexed.h File Reference . . . . .	1299
9.47.1 Detailed Description . . . . .	1299
9.48 inifcns.cpp File Reference . . . . .	1300
9.48.1 Detailed Description . . . . .	1303
9.49 inifcns.h File Reference . . . . .	1303
9.49.1 Detailed Description . . . . .	1304
9.50 inifcns_elliptic.cpp File Reference . . . . .	1304
9.50.1 Detailed Description . . . . .	1305
9.51 inifcns_gamma.cpp File Reference . . . . .	1306
9.51.1 Detailed Description . . . . .	1307
9.52 inifcns_nstdsums.cpp File Reference . . . . .	1307
9.52.1 Detailed Description . . . . .	1308
9.53 inifcns_trans.cpp File Reference . . . . .	1309

9.53.1 Detailed Description	1312
9.54 integral.cpp File Reference	1312
9.54.1 Detailed Description	1313
9.55 integral.h File Reference	1313
9.55.1 Detailed Description	1313
9.56 integration_kernel.cpp File Reference	1313
9.56.1 Detailed Description	1315
9.57 integration_kernel.h File Reference	1315
9.57.1 Detailed Description	1316
9.58 lst.cpp File Reference	1316
9.58.1 Detailed Description	1317
9.59 lst.h File Reference	1317
9.59.1 Detailed Description	1317
9.60 matrix.cpp File Reference	1317
9.60.1 Detailed Description	1318
9.61 matrix.h File Reference	1318
9.61.1 Detailed Description	1319
9.62 mul.cpp File Reference	1320
9.62.1 Detailed Description	1320
9.63 mul.h File Reference	1320
9.63.1 Detailed Description	1321
9.64 ncmul.cpp File Reference	1321
9.64.1 Detailed Description	1322
9.65 ncmul.h File Reference	1322
9.65.1 Detailed Description	1322
9.66 normal.cpp File Reference	1323
9.66.1 Detailed Description	1325
9.66.2 Macro Definition Documentation	1325
9.66.2.1 FAST_COMPARE	1325
9.66.2.2 USE_REMEMBER	1325
9.66.2.3 USE_TRIAL_DIVISION	1325
9.66.2.4 STATISTICS	1325
9.67 normal.h File Reference	1325
9.67.1 Detailed Description	1326
9.68 numeric.cpp File Reference	1327
9.68.1 Detailed Description	1330
9.69 numeric.h File Reference	1330
9.69.1 Detailed Description	1333
9.70 operators.cpp File Reference	1333
9.70.1 Detailed Description	1334
9.71 operators.h File Reference	1335
9.71.1 Detailed Description	1336

9.72 power.cpp File Reference	1336
9.72.1 Detailed Description	1337
9.73 power.h File Reference	1337
9.73.1 Detailed Description	1337
9.74 print.cpp File Reference	1338
9.74.1 Detailed Description	1338
9.75 print.h File Reference	1338
9.75.1 Detailed Description	1339
9.75.2 Macro Definition Documentation	1340
9.75.2.1 GINAC_DECLARE_PRINT_CONTEXT_COMMON	1340
9.75.2.2 GINAC_DECLARE_PRINT_CONTEXT_BASE	1340
9.75.2.3 GINAC_DECLARE_PRINT_CONTEXT	1340
9.75.2.4 GINAC_IMPLEMENT_PRINT_CONTEXT	1341
9.76 pseries.cpp File Reference	1341
9.76.1 Detailed Description	1341
9.77 pseries.h File Reference	1342
9.77.1 Detailed Description	1342
9.78 ptr.h File Reference	1342
9.78.1 Detailed Description	1343
9.79 registrar.cpp File Reference	1343
9.79.1 Detailed Description	1343
9.80 registrar.h File Reference	1343
9.80.1 Detailed Description	1344
9.80.2 Macro Definition Documentation	1344
9.80.2.1 GINAC_DECLARE_REGISTERED_CLASS_COMMON	1344
9.80.2.2 GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS	1345
9.80.2.3 GINAC_DECLARE_REGISTERED_CLASS	1345
9.80.2.4 GINAC_IMPLEMENT_REGISTERED_CLASS	1345
9.80.2.5 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT	1346
9.80.2.6 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T	1346
9.81 relational.cpp File Reference	1346
9.81.1 Detailed Description	1347
9.82 relational.h File Reference	1347
9.82.1 Detailed Description	1347
9.83 remember.cpp File Reference	1347
9.83.1 Detailed Description	1348
9.84 remember.h File Reference	1348
9.84.1 Detailed Description	1348
9.85 structure.h File Reference	1348
9.85.1 Detailed Description	1349
9.86 symbol.cpp File Reference	1349
9.86.1 Detailed Description	1350

9.87 symbol.h File Reference	1350
9.87.1 Detailed Description	1351
9.88 symmetry.cpp File Reference	1351
9.88.1 Detailed Description	1352
9.89 symmetry.h File Reference	1352
9.89.1 Detailed Description	1353
9.90 tensor.cpp File Reference	1353
9.90.1 Detailed Description	1354
9.91 tensor.h File Reference	1354
9.91.1 Detailed Description	1355
9.92 utils.cpp File Reference	1355
9.92.1 Detailed Description	1358
9.93 utils.h File Reference	1358
9.93.1 Detailed Description	1359
9.93.2 Macro Definition Documentation	1359
9.93.2.1 DEFAULT_CTOR	1359
9.93.2.2 DEFAULT_COMPARE	1359
9.93.2.3 DEFAULT_PRINT	1360
9.93.2.4 DEFAULT_PRINT_LATEX	1360
9.94 utils_multi_iterator.h File Reference	1360
9.94.1 Detailed Description	1361
9.95 version.h File Reference	1361
9.95.1 Detailed Description	1362
9.95.2 Macro Definition Documentation	1362
9.95.2.1 GINACLIB_MAJOR_VERSION	1362
9.95.2.2 GINACLIB_MINOR_VERSION	1362
9.95.2.3 GINACLIB_MICRO_VERSION	1362
9.95.2.4 GINAC_LT_CURRENT	1362
9.95.2.5 GINAC_LT_REVISION	1362
9.95.2.6 GINAC_LT_AGE	1363
9.95.2.7 GINACLIB_ARCHIVE_VERSION	1363
9.95.2.8 GINACLIB_ARCHIVE_AGE	1363
9.95.2.9 GINACLIB_STR_HELPER	1363
9.95.2.10 GINACLIB_STR	1363
9.95.2.11 GINACLIB_VERSION	1363
9.96 wildcard.cpp File Reference	1363
9.96.1 Detailed Description	1364
9.97 wildcard.h File Reference	1364
9.97.1 Detailed Description	1364

# Chapter 1

## Directory Hierarchy

### 1.1 Directories

ginac	23
add.cpp	1245
add.h	1246
archive.cpp	1246
archive.h	1247
assertion.h	1249
basic.cpp	1251
basic.h	1252
class_info.h	1253
clifford.cpp	1254
clifford.h	1256
color.cpp	1258
color.h	1260
compiler.h	1261
constant.cpp	1262
constant.h	1263
container.h	1264
crc32.h	1264
ex.cpp	1265
ex.h	1265
excompiler.cpp	1268
excompiler.h	1268
expair.cpp	1269
expair.h	1270
expairseq.cpp	1271
expairseq.h	1271
exprseq.cpp	1272
exprseq.h	1273
factor.cpp	1273
factor.h	1275
fail.cpp	1275
fail.h	1276
fderivative.cpp	1276
fderivative.h	1277
flags.h	1277
function.cpp	1278
function.h	1279
ginac.h	1290

Gt.cpp	1291
Gt.h	1292
Gt_helpers.cpp	1292
Gt_helpers.h	1293
hash_map.h	1294
hash_seed.h	1294
idx.cpp	1295
idx.h	1296
indexed.cpp	1297
indexed.h	1299
inifcns.cpp	1300
inifcns.h	1303
inifcns_elliptic.cpp	1304
inifcns_gamma.cpp	1306
inifcns_nstdsums.cpp	1307
inifcns_trans.cpp	1309
integral.cpp	1312
integral.h	1313
integration_kernel.cpp	1313
integration_kernel.h	1315
lst.cpp	1316
lst.h	1317
matrix.cpp	1317
matrix.h	1318
mul.cpp	1320
mul.h	1320
ncmul.cpp	1321
ncmul.h	1322
normal.cpp	1323
normal.h	1325
numeric.cpp	1327
numeric.h	1330
operators.cpp	1333
operators.h	1335
power.cpp	1336
power.h	1337
print.cpp	1338
print.h	1338
pseries.cpp	1341
pseries.h	1342
ptr.h	1342
registrar.cpp	1343
registrar.h	1343
relational.cpp	1346
relational.h	1347
remember.cpp	1347
remember.h	1348
structure.h	1348
symbol.cpp	1349
symbol.h	1350
symmetry.cpp	1351
symmetry.h	1352
tensor.cpp	1353
tensor.h	1354
utils.cpp	1355
utils.h	1358
utils_multi_iterator.h	1360
version.h	1361

wildcard.cpp	1363
wildcard.h	1364





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">GiNaC</a>	29
<a href="#">GiNaC::Gt_detail</a>	300
<a href="#">GiNaC::internal</a>	302
<a href="#">std</a>	302



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

GiNaC::internal::_iter_rep	305
GiNaC::_numeric_digits	307
GiNaC::archive	327
GiNaC::archive_node	334
GiNaC::archive_node::archive_node_cit_range	344
GiNaC::archive::archived_ex	345
GiNaC::basic_multi_iterator< T >	382
GiNaC::multi_iterator_counter< T >	859
GiNaC::multi_iterator_counter_indv< T >	862
GiNaC::multi_iterator_ordered< T >	866
GiNaC::multi_iterator_ordered_eq< T >	870
GiNaC::multi_iterator_ordered_eq_indv< T >	874
GiNaC::multi_iterator_permutation< T >	878
GiNaC::multi_iterator_shuffle< T >	882
GiNaC::multi_iterator_shuffle_prime< T >	886
GiNaC::basic_partition_generator	388
GiNaC::partition_generator	941
GiNaC::partition_with_zero_parts_generator	943
GiNaC::class_info< OPT >	390
GiNaC::compare_all_equal< T >	425
GiNaC::compare_bitwise< T >	426
GiNaC::compare_std_less< T >	427
ComparisonPolicy	
GiNaC::structure< T, ComparisonPolicy >	1114
GiNaC::composition_generator	428
GiNaC::const_iterator	430
GiNaC::const_postorder_iterator	437
GiNaC::const_preorder_iterator	440
GiNaC::container_storage< C >	469
GiNaC::container< std::vector >	454
GiNaC::function	630
GiNaC::fderivative	616
GiNaC::indexed	727
GiNaC::clifford	394

GiNaC::color	413
GiNaC::ncmul	898
GiNaC::container< std::list >	454
GiNaC::container< C >	454
GiNaC::composition_generator::coolmulti	473
GiNaC::determinant_algo	477
GiNaC::do_taylor	504
GiNaC::domain	504
std::domain_error	
GiNaC::pole_error	974
GiNaC::dunno	505
GiNaC::composition_generator::coolmulti::element	533
std::equal_to< GiNaC::ex >	543
GiNaC::error_and_integral	544
GiNaC::error_and_integral_is_less	545
GiNaC::ex	550
GiNaC::ex_base_is_less	582
GiNaC::ex_is_equal	582
GiNaC::ex_is_less	583
GiNaC::ex_swap	583
GiNaC::expair	584
GiNaC::expair_is_less	588
GiNaC::expair_rest_is_less	589
GiNaC::expair_swap	590
GiNaC::expand_options	610
GiNaC::factor_options	611
GiNaC::function_options	650
GiNaC::G2_SERIAL	692
GiNaC::G3_SERIAL	693
GiNaC::gcd_options	693
GiNaC::gcdheu_failed	694
GiNaC::has_distance< T >	710
GiNaC::has_options	712
std::hash< GiNaC::ex >	713
GiNaC::idx_is_equal_ignore_dim	726
GiNaC::info_flags	746
GiNaC::is_not_a_clifford	768
GiNaC::is_summation_idx	769
GiNaC::iterated_integral2_SERIAL	769
GiNaC::iterated_integral3_SERIAL	770
GiNaC::Gt::kernel	770
GiNaC::lanczos_coeffs	790
std::less< GiNaC::ptr< T > >	792
GiNaC::library_init	793
std::list	
GiNaC::remember_table_list	1085
GiNaC::make_flat_inserter	795
GiNaC::map_function	797
GiNaC::derivative_map_function	475
GiNaC::eval_integ_map_function	546
GiNaC::evalf_map_function	547
GiNaC::evalm_map_function	549
GiNaC::expand_map_function	608
GiNaC::normal_map_function	911
GiNaC::pointer_to_map_function	952
GiNaC::pointer_to_map_function_1arg< T1 >	954
GiNaC::pointer_to_map_function_2args< T1, T2 >	957
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >	959

GiNaC::pointer_to_member_to_map_function< C > . . . . .	962
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > . . . . .	965
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > . . . . .	968
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > . . . . .	971
GiNaC::basic_partition_generator::mpartition2 . . . . .	839
GiNaC::op0_is_equal . . . . .	940
GiNaC::Gt_detail::pathintegral_term . . . . .	946
GiNaC::Gt_detail::pathintegral_term_list . . . . .	950
GiNaC::print_context . . . . .	999
GiNaC::print_csrc . . . . .	1003
GiNaC::print_csrc_cl_N . . . . .	1004
GiNaC::print_csrc_double . . . . .	1006
GiNaC::print_csrc_float . . . . .	1008
GiNaC::print_dflt . . . . .	1010
GiNaC::print_latex . . . . .	1015
GiNaC::print_python . . . . .	1022
GiNaC::print_python_repr . . . . .	1024
GiNaC::print_tree . . . . .	1025
GiNaC::print_context_options . . . . .	1001
GiNaC::print_functor . . . . .	1011
GiNaC::print_functor_impl . . . . .	1014
GiNaC::print_memfun_handler< T, C > . . . . .	1017
GiNaC::print_ptrfun_handler< T, C > . . . . .	1020
GiNaC::print_options . . . . .	1019
GiNaC::archive_node::property . . . . .	1027
GiNaC::archive_node::property_info . . . . .	1029
GiNaC::psi1_SERIAL . . . . .	1047
GiNaC::psi2_SERIAL . . . . .	1048
GiNaC::ptr< T > . . . . .	1048
GiNaC::refcounted . . . . .	1060
GiNaC::basic . . . . .	346
GiNaC::container< std::vector > . . . . .	454
GiNaC::container< std::list > . . . . .	454
GiNaC::Gt . . . . .	694
GiNaC::constant . . . . .	443
GiNaC::container< C > . . . . .	454
GiNaC::expairseq . . . . .	590
GiNaC::add . . . . .	310
GiNaC::mul . . . . .	840
GiNaC::fail . . . . .	612
GiNaC::idx . . . . .	713
GiNaC::varidx . . . . .	1227
GiNaC::spinidx . . . . .	1093
GiNaC::integral . . . . .	747
GiNaC::integration_kernel . . . . .	759
GiNaC::ELi_kernel . . . . .	535
GiNaC::Ebar_kernel . . . . .	505
GiNaC::Eisenstein_h_kernel . . . . .	513
GiNaC::Eisenstein_kernel . . . . .	524
GiNaC::Kronecker_dtau_kernel . . . . .	773
GiNaC::Kronecker_dz_kernel . . . . .	782
GiNaC::basic_log_kernel . . . . .	376
GiNaC::modular_form_kernel . . . . .	830
GiNaC::multiple_polylog_kernel . . . . .	890
GiNaC::user_defined_kernel . . . . .	1219
GiNaC::matrix . . . . .	800
GiNaC::numeric . . . . .	912

GiNaC::power	982
GiNaC::pseries	1030
GiNaC::relational	1066
GiNaC::structure< T, ComparisonPolicy >	1114
GiNaC::symbol	1160
GiNaC::realsymbol	1053
GiNaC::possymbol	976
GiNaC::symmetry	1173
GiNaC::tensor	1206
GiNaC::cliffordunit	408
GiNaC::diracgamma	478
GiNaC::diracgamma5	484
GiNaC::diracgammaL	489
GiNaC::diracgammaR	494
GiNaC::diracone	499
GiNaC::su3d	1133
GiNaC::su3f	1138
GiNaC::su3one	1144
GiNaC::su3t	1149
GiNaC::tensdelta	1187
GiNaC::tensepsilon	1193
GiNaC::tensmetric	1200
GiNaC::minkmetric	823
GiNaC::spinmetric	1104
GiNaC::wildcard	1236
GiNaC::registered_class_options	1063
GiNaC::remember_strategies	1077
GiNaC::remember_table_entry	1082
GiNaC::return_type_t	1087
GiNaC::return_types	1088
GiNaC::relational::safe_bool_helper	1089
GiNaC::scalar_products	1089
GiNaC::series_options	1092
GiNaC::solve_algo	1092
GiNaC::spmapkey	1110
GiNaC::status_flags	1113
GiNaC::subs_options	1154
GiNaC::sy_is_less	1155
GiNaC::sy_swap	1155
GiNaC::sym_desc	1157
GiNaC::symbolset	1172
GiNaC::symminfo	1183
GiNaC::symminfo_is_less_by_orig	1186
GiNaC::symminfo_is_less_by_symmterm	1186
GiNaC::terminfo	1211
GiNaC::terminfo_is_less	1213
GiNaC::Gt_detail::TransformExpressionWithCache< Type >	1213
GiNaC::class_info< OPT >tree_node	1216
GiNaC::unarchive_table_t	1218
std::vector	
GiNaC::remember_table	1078
GiNaC::visitor	1236
GiNaC::zeta1_SERIAL	1243
GiNaC::zeta2_SERIAL	1243

## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">GiNaC::internal::_iter_rep</a>	305
<a href="#">GiNaC::_numeric_digits</a>	
This class is used to instantiate a global singleton object Digits which behaves just like Maple's	
Digits	307
<a href="#">GiNaC::add</a>	
Sum of expressions	310
<a href="#">GiNaC::archive</a>	
This class holds archived versions of <a href="#">GiNaC</a> expressions (class <a href="#">ex</a> )	327
<a href="#">GiNaC::archive_node</a>	
This class stores all properties needed to record/retrieve the state of one object of class <a href="#">basic</a>	
(or a derived class)	334
<a href="#">GiNaC::archive_node::archive_node_cit_range</a>	344
<a href="#">GiNaC::archive::archived_ex</a>	
Archived expression descriptor	345
<a href="#">GiNaC::basic</a>	
This class is the ABC (abstract base class) of <a href="#">GiNaC</a> 's class hierarchy	346
<a href="#">GiNaC::basic_log_kernel</a>	
The basic integration kernel with a logarithmic singularity at the origin	376
<a href="#">GiNaC::basic_multi_iterator&lt; T &gt;</a>	
Basic_multi_iterator is a base class	382
<a href="#">GiNaC::basic_partition_generator</a>	
Base class for generating all bounded combinatorial partitions of an integer n with exactly m	
parts in non-decreasing order	388
<a href="#">GiNaC::class_info&lt; OPT &gt;</a>	390
<a href="#">GiNaC::clifford</a>	
This class holds an object representing an element of the Clifford algebra (the Dirac gamma	
matrices)	394
<a href="#">GiNaC::cliffordunit</a>	
This class represents the Clifford algebra generators (units)	408
<a href="#">GiNaC::color</a>	
This class holds a generator T_a or the unity element of the Lie algebra of SU(3), as used for	
calculations in quantum chromodynamics	413
<a href="#">GiNaC::compare_all_equal&lt; T &gt;</a>	
Comparison policy: all structures of one type are equal	425
<a href="#">GiNaC::compare_bitwise&lt; T &gt;</a>	
Comparison policy: use bit-wise comparison to compare structures	426

<a href="#">GiNaC::compare_std_less&lt; T &gt;</a>	
Comparison policy: use <code>std::equal_to</code> / <code>std::less</code> (defaults to operators <code>==</code> and <code>&lt;</code> ) to compare structures . . . . .	427
<a href="#">GiNaC::composition_generator</a>	
Generate all compositions of a partition of an integer $n$ , starting with the compositions which has non-decreasing order . . . . .	428
<a href="#">GiNaC::const_iterator</a> . . . . .	430
<a href="#">GiNaC::const_postorder_iterator</a> . . . . .	437
<a href="#">GiNaC::const_preorder_iterator</a> . . . . .	440
<a href="#">GiNaC::constant</a>	
This class holds constants, symbols with specific numerical value . . . . .	443
<a href="#">GiNaC::container&lt; C &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers . . . . .	454
<a href="#">GiNaC::container_storage&lt; C &gt;</a>	
Helper template for encapsulating the <a href="#">reserve()</a> mechanics of STL containers . . . . .	469
<a href="#">GiNaC::composition_generator::coolmulti</a> . . . . .	473
<a href="#">GiNaC::derivative_map_function</a>	
Function object to be applied by <a href="#">basic::derivative()</a> . . . . .	475
<a href="#">GiNaC::determinant_algo</a>	
Switch to control algorithm for determinant computation . . . . .	477
<a href="#">GiNaC::diracgamma</a>	
This class represents the Dirac gamma Lorentz vector . . . . .	478
<a href="#">GiNaC::diracgamma5</a>	
This class represents the Dirac gamma5 object which anticommutes with all other gammas . . . . .	484
<a href="#">GiNaC::diracgammaL</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1-\text{gamma}5)$ . . . . .	489
<a href="#">GiNaC::diracgammaR</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1+\text{gamma}5)$ . . . . .	494
<a href="#">GiNaC::diracone</a>	
This class represents the Clifford algebra unity element . . . . .	499
<a href="#">GiNaC::do_taylor</a>	
Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe . . . . .	504
<a href="#">GiNaC::domain</a>	
Domain of an object . . . . .	504
<a href="#">GiNaC::dunno</a>	
Exception class thrown by functions to signal unimplemented functionality so the expression may just be <code>.hold()</code> . . . . .	505
<a href="#">GiNaC::Ebar_kernel</a>	
The Ebar-kernel . . . . .	505
<a href="#">GiNaC::Eisenstein_h_kernel</a>	
The kernel corresponding to the Eisenstein series $h_{k,N,r,s}(\tau)$ . . . . .	513
<a href="#">GiNaC::Eisenstein_kernel</a>	
The kernel corresponding to the Eisenstein series $E_{k,N,a,b,K}(\tau)$ . . . . .	524
<a href="#">GiNaC::composition_generator::coolmulti::element</a> . . . . .	533
<a href="#">GiNaC::ELi_kernel</a>	
The ELi-kernel . . . . .	535
<a href="#">std::equal_to&lt; GiNaC::ex &gt;</a>	
Specialization of <code>std::equal_to()</code> for <code>ex</code> objects . . . . .	543
<a href="#">GiNaC::error_and_integral</a> . . . . .	544
<a href="#">GiNaC::error_and_integral_is_less</a> . . . . .	545
<a href="#">GiNaC::eval_integ_map_function</a>	
Function object to be applied by <a href="#">basic::eval_integ()</a> . . . . .	546
<a href="#">GiNaC::evalf_map_function</a>	
Function object to be applied by <a href="#">basic::evalf()</a> . . . . .	547
<a href="#">GiNaC::evalm_map_function</a>	
Function object to be applied by <a href="#">basic::evalm()</a> . . . . .	549



<a href="#">GiNaC::ex</a>	
Lightweight wrapper for <a href="#">GiNaC</a> 's symbolic objects	550
<a href="#">GiNaC::ex_base_is_less</a>	582
<a href="#">GiNaC::ex_is_equal</a>	582
<a href="#">GiNaC::ex_is_less</a>	583
<a href="#">GiNaC::ex_swap</a>	583
<a href="#">GiNaC::expair</a>	
A pair of expressions	584
<a href="#">GiNaC::expair_is_less</a>	
Function object for insertion into third argument of STL's sort() etc	588
<a href="#">GiNaC::expair_rest_is_less</a>	
Function object not caring about the numerical coefficients for insertion into third argument of STL's sort()	589
<a href="#">GiNaC::expair_swap</a>	590
<a href="#">GiNaC::expairseq</a>	
A sequence of class expair	590
<a href="#">GiNaC::expand_map_function</a>	
Function object to be applied by <a href="#">basic::expand()</a>	608
<a href="#">GiNaC::expand_options</a>	
Flags to control the behavior of <a href="#">expand()</a>	610
<a href="#">GiNaC::factor_options</a>	
Flags to control the polynomial factorization	611
<a href="#">GiNaC::fail</a>	612
<a href="#">GiNaC::fderivative</a>	
This class represents the (abstract) derivative of a symbolic function	616
<a href="#">GiNaC::function</a>	
The class function is used to implement builtin functions like sin, cos... and user defined functions	630
<a href="#">GiNaC::function_options</a>	650
<a href="#">GiNaC::G2_SERIAL</a>	
Generalized multiple polylogarithm	692
<a href="#">GiNaC::G3_SERIAL</a>	
Generalized multiple polylogarithm with explicit imaginary parts	693
<a href="#">GiNaC::gcd_options</a>	
Flags to control the behavior of <a href="#">gcd()</a> and friends	693
<a href="#">GiNaC::gcdheu_failed</a>	
Exception thrown by <a href="#">heur_gcd()</a> to signal failure	694
<a href="#">GiNaC::Gt</a>	
Elliptic Multiple Polylogarithm	694
<a href="#">GiNaC::has_distance&lt; T &gt;</a>	
SFINAE test for distance	710
<a href="#">GiNaC::has_options</a>	
Flags to control the behavior of <a href="#">has()</a>	712
<a href="#">std::hash&lt; GiNaC::ex &gt;</a>	
Specialization of std::hash() for ex objects	713
<a href="#">GiNaC::idx</a>	
This class holds one index of an indexed object	713
<a href="#">GiNaC::idx_is_equal_ignore_dim</a>	726
<a href="#">GiNaC::indexed</a>	
This class holds an indexed expression	727
<a href="#">GiNaC::info_flags</a>	
Possible attributes an object can have	746
<a href="#">GiNaC::integral</a>	
Symbolic integral	747
<a href="#">GiNaC::integration_kernel</a>	
The base class for integration kernels for iterated integrals	759
<a href="#">GiNaC::is_not_a_clifford</a>	
Predicate for finding non-clifford objects	768
<a href="#">GiNaC::is_summation_idx</a>	769

<a href="#">GiNaC::iterated_integral2_SERIAL</a>	
Complete elliptic integral of the first kind	769
<a href="#">GiNaC::iterated_integral3_SERIAL</a>	
Iterated integral with explicit truncation	770
<a href="#">GiNaC::Gt::kernel</a>	770
<a href="#">GiNaC::Kronecker_dtau_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n)}(z_j, K\tau)$ in $\tau$ (or equivalently in $\bar{q}$ )	773
<a href="#">GiNaC::Kronecker_dz_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n-1)}(z - z_j, K\tau)$ in $z$	782
<a href="#">GiNaC::lanczos_coeffs</a>	790
<a href="#">std::less&lt; GiNaC::ptr&lt; T &gt; &gt;</a>	
Specialization of <code>std::less</code> for <code>ptr&lt;T&gt;</code> to enable ordering of <code>ptr&lt;T&gt;</code> objects (e.g	792
<a href="#">GiNaC::library_init</a>	
Helper class to initialize the library	793
<a href="#">GiNaC::make_flat_inserter</a>	
Class to handle the renaming of dummy indices	795
<a href="#">GiNaC::map_function</a>	
Function object for <code>map()</code>	797
<a href="#">GiNaC::matrix</a>	
Symbolic matrices	800
<a href="#">GiNaC::minkmetric</a>	
This class represents a Minkowski metric tensor	823
<a href="#">GiNaC::modular_form_kernel</a>	
A kernel corresponding to a polynomial in Eisenstein series	830
<a href="#">GiNaC::basic_partition_generator::mpartition2</a>	839
<a href="#">GiNaC::mul</a>	
Product of expressions	840
<a href="#">GiNaC::multi_iterator_counter&lt; T &gt;</a>	
The class <code>multi_iterator_counter</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	859
<a href="#">GiNaC::multi_iterator_counter_indv&lt; T &gt;</a>	
The class <code>multi_iterator_counter_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	862
<a href="#">GiNaC::multi_iterator_ordered&lt; T &gt;</a>	
The class <code>multi_iterator_ordered</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	866
<a href="#">GiNaC::multi_iterator_ordered_eq&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	870
<a href="#">GiNaC::multi_iterator_ordered_eq_indv&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	874
<a href="#">GiNaC::multi_iterator_permutation&lt; T &gt;</a>	
The class <code>multi_iterator_permutation</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , for which	878
<a href="#">GiNaC::multi_iterator_shuffle&lt; T &gt;</a>	
The class <code>multi_iterator_shuffle</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b	882
<a href="#">GiNaC::multi_iterator_shuffle_prime&lt; T &gt;</a>	
The class <code>multi_iterator_shuffle_prime</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b, excluding the first one (a,b)	886
<a href="#">GiNaC::multiple_polylog_kernel</a>	
The integration kernel for multiple polylogarithms	890
<a href="#">GiNaC::ncmul</a>	
Non-commutative product of expressions	898
<a href="#">GiNaC::normal_map_function</a>	
Function object to be applied by <code>basic::normal()</code>	911
<a href="#">GiNaC::numeric</a>	
This class is a wrapper around CLN-numbers within the <code>GiNaC</code> class hierarchy	912
<a href="#">GiNaC::op0_is_equal</a>	940
<a href="#">GiNaC::partition_generator</a>	
Generate all bounded combinatorial partitions of an integer $n$ with exactly $m$ parts (not including zero parts) in non-decreasing order	941

<a href="#">GiNaC::partition_with_zero_parts_generator</a>	
Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order	943
<a href="#">GiNaC::Gt_detail::pathintegral_term</a>	946
<a href="#">GiNaC::Gt_detail::pathintegral_term_list</a>	950
<a href="#">GiNaC::pointer_to_map_function</a>	952
<a href="#">GiNaC::pointer_to_map_function_1arg&lt; T1 &gt;</a>	954
<a href="#">GiNaC::pointer_to_map_function_2args&lt; T1, T2 &gt;</a>	957
<a href="#">GiNaC::pointer_to_map_function_3args&lt; T1, T2, T3 &gt;</a>	959
<a href="#">GiNaC::pointer_to_member_to_map_function&lt; C &gt;</a>	962
<a href="#">GiNaC::pointer_to_member_to_map_function_1arg&lt; C, T1 &gt;</a>	965
<a href="#">GiNaC::pointer_to_member_to_map_function_2args&lt; C, T1, T2 &gt;</a>	968
<a href="#">GiNaC::pointer_to_member_to_map_function_3args&lt; C, T1, T2, T3 &gt;</a>	971
<a href="#">GiNaC::pole_error</a>	
Exception class thrown when a singularity is encountered	974
<a href="#">GiNaC::possymbol</a>	
Specialization of symbol to real positive domain	976
<a href="#">GiNaC::power</a>	
This class holds a two-component object, a basis and and exponent representing exponentiation	982
<a href="#">GiNaC::print_context</a>	
Base class for print_contexts	999
<a href="#">GiNaC::print_context_options</a>	
This class stores information about a registered <a href="#">print_context</a> class	1001
<a href="#">GiNaC::print_csrc</a>	
Base context for C source output	1003
<a href="#">GiNaC::print_csrc_cl_N</a>	
Context for C source output using CLN numbers	1004
<a href="#">GiNaC::print_csrc_double</a>	
Context for C source output using double precision	1006
<a href="#">GiNaC::print_csrc_float</a>	
Context for C source output using float precision	1008
<a href="#">GiNaC::print_dflt</a>	
Context for default (ginsh-parsable) output	1010
<a href="#">GiNaC::print_functor</a>	
This class represents a print method for a certain algebraic class and <a href="#">print_context</a> type	1011
<a href="#">GiNaC::print_functor_impl</a>	
Base class for <a href="#">print_functor</a> handlers	1014
<a href="#">GiNaC::print_latex</a>	
Context for latex-parsable output	1015
<a href="#">GiNaC::print_memfun_handler&lt; T, C &gt;</a>	
Print_functor handler for member functions of class T, context type C	1017
<a href="#">GiNaC::print_options</a>	
Flags to control the behavior of a <a href="#">print_context</a>	1019
<a href="#">GiNaC::print_ptrfun_handler&lt; T, C &gt;</a>	
Print_functor handler for pointer-to-functions of class T, context type C	1020
<a href="#">GiNaC::print_python</a>	
Context for python pretty-print output	1022
<a href="#">GiNaC::print_python_repr</a>	
Context for python-parsable output	1024
<a href="#">GiNaC::print_tree</a>	
Context for tree-like output for debugging	1025
<a href="#">GiNaC::archive_node::property</a>	
Archived property (data type, name and associated data)	1027
<a href="#">GiNaC::archive_node::property_info</a>	
Information about a stored property	1029
<a href="#">GiNaC::pseries</a>	
This class holds a extended truncated power series (positive and negative integer powers)	1030

<a href="#">GiNaC::psi1_SERIAL</a>	
Polylogarithm and multiple polylogarithm . . . . .	1047
<a href="#">GiNaC::psi2_SERIAL</a>	
Derivatives of Psi-function (aka polygamma-functions) . . . . .	1048
<a href="#">GiNaC::ptr&lt; T &gt;</a>	
Class of (intrusively) reference-counted pointers that support copy-on-write semantics . . . . .	1048
<a href="#">GiNaC::realsymbol</a>	
Specialization of symbol to real domain . . . . .	1053
<a href="#">GiNaC::refcounted</a>	
Base class for reference-counted objects . . . . .	1060
<a href="#">GiNaC::registered_class_options</a>	
This class stores information about a registered <a href="#">GiNaC</a> class . . . . .	1063
<a href="#">GiNaC::relational</a>	
This class holds a relation consisting of two expressions and a logical relation between them . . . . .	1066
<a href="#">GiNaC::remember_strategies</a>	
Strategies how to clean up the function remember cache . . . . .	1077
<a href="#">GiNaC::remember_table</a>	
The remember table is organized like an n-fold associative cache in a microprocessor . . . . .	1078
<a href="#">GiNaC::remember_table_entry</a>	
A single entry in the remember table of a function . . . . .	1082
<a href="#">GiNaC::remember_table_list</a>	
A list of entries in the remember table having some least significant bits of the hashvalue in common . . . . .	1085
<a href="#">GiNaC::return_type_t</a>	
To distinguish between different kinds of non-commutative objects . . . . .	1087
<a href="#">GiNaC::return_types</a>	
. . . . .	1088
<a href="#">GiNaC::relational::safe_bool_helper</a>	
. . . . .	1089
<a href="#">GiNaC::scalar_products</a>	
Helper class for storing information about known scalar products which are to be automatically replaced by <a href="#">simplify_indexed()</a> . . . . .	1089
<a href="#">GiNaC::series_options</a>	
Flags to control series expansion . . . . .	1092
<a href="#">GiNaC::solve_algo</a>	
Switch to control algorithm for linear system solving . . . . .	1092
<a href="#">GiNaC::spinidx</a>	
This class holds a spinor index that can be dotted or undotted and that also has a variance . . . . .	1093
<a href="#">GiNaC::spinmetric</a>	
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors . . . . .	1104
<a href="#">GiNaC::spmapkey</a>	
. . . . .	1110
<a href="#">GiNaC::status_flags</a>	
Flags to store information about the state of an object . . . . .	1113
<a href="#">GiNaC::structure&lt; T, ComparisonPolicy &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of C++ structures . . . . .	1114
<a href="#">GiNaC::su3d</a>	
This class represents the tensor of symmetric su(3) structure constants . . . . .	1133
<a href="#">GiNaC::su3f</a>	
This class represents the tensor of antisymmetric su(3) structure constants . . . . .	1138
<a href="#">GiNaC::su3one</a>	
This class represents the su(3) unity element . . . . .	1144
<a href="#">GiNaC::su3t</a>	
This class represents an su(3) generator . . . . .	1149
<a href="#">GiNaC::subs_options</a>	
Flags to control the behavior of <a href="#">subs()</a> . . . . .	1154
<a href="#">GiNaC::sy_is_less</a>	
. . . . .	1155
<a href="#">GiNaC::sy_swap</a>	
. . . . .	1155

<a href="#">GiNaC::sym_desc</a>	
This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b" . . . . .	1157
<a href="#">GiNaC::symbol</a>	
Basic CAS symbol . . . . .	1160
<a href="#">GiNaC::symbolset</a> . . . . .	1172
<a href="#">GiNaC::symmetry</a>	
This class describes the symmetry of a group of indices . . . . .	1173
<a href="#">GiNaC::symminfo</a>	
This structure stores the individual symmetrized terms obtained during the simplification of sums	1183
<a href="#">GiNaC::symminfo_is_less_by_orig</a> . . . . .	1186
<a href="#">GiNaC::symminfo_is_less_by_symmterm</a> . . . . .	1186
<a href="#">GiNaC::tensdelta</a>	
This class represents the delta tensor . . . . .	1187
<a href="#">GiNaC::tensepsilon</a>	
This class represents the totally antisymmetric epsilon tensor . . . . .	1193
<a href="#">GiNaC::tensmetric</a>	
This class represents a general metric tensor which can be used to raise/lower indices . . . .	1200
<a href="#">GiNaC::tensor</a>	
This class holds one of <a href="#">GiNaC</a> 's predefined special tensors such as the delta and the metric tensors . . . . .	1206
<a href="#">GiNaC::terminfo</a>	
This structure stores the original and symmetrized versions of terms obtained during the simplification of sums . . . . .	1211
<a href="#">GiNaC::terminfo_is_less</a> . . . . .	1213
<a href="#">GiNaC::Gt_detail::TransformExpressionWithCache&lt; Type &gt;</a> . . . . .	1213
<a href="#">GiNaC::class_info&lt; OPT &gt;::tree_node</a> . . . . .	1216
<a href="#">GiNaC::unarchive_table_t</a> . . . . .	1218
<a href="#">GiNaC::user_defined_kernel</a>	
A user-defined integration kernel . . . . .	1219
<a href="#">GiNaC::varidx</a>	
This class holds an index with a variance (co- or contravariant) . . . . .	1227
<a href="#">GiNaC::visitor</a>	
Degenerate base class for visitors . . . . .	1236
<a href="#">GiNaC::wildcard</a>	
This class acts as a wildcard for <a href="#">subs()</a> , <a href="#">match()</a> , <a href="#">has()</a> and <a href="#">find()</a> . . . . .	1236
<a href="#">GiNaC::zeta1_SERIAL</a>	
Complex conjugate . . . . .	1243
<a href="#">GiNaC::zeta2_SERIAL</a>	
Alternating Euler sum or colored MZV . . . . .	1243



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">add.cpp</a>	Implementation of <a href="#">GiNaC</a> 's sums of expressions . . . . .	1245
<a href="#">add.h</a>	Interface to <a href="#">GiNaC</a> 's sums of expressions . . . . .	1246
<a href="#">archive.cpp</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	1246
<a href="#">archive.h</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	1247
<a href="#">assertion.h</a>	Assertion macro definition . . . . .	1249
<a href="#">basic.cpp</a>	Implementation of <a href="#">GiNaC</a> 's ABC . . . . .	1251
<a href="#">basic.h</a>	Interface to <a href="#">GiNaC</a> 's ABC . . . . .	1252
<a href="#">class_info.h</a>	Helper templates to provide per-class information for class hierarchies . . . . .	1253
<a href="#">clifford.cpp</a>	Implementation of <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1254
<a href="#">clifford.h</a>	Interface to <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1256
<a href="#">color.cpp</a>	Implementation of <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1258
<a href="#">color.h</a>	Interface to <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1260
<a href="#">compiler.h</a>	Definition of optimizing macros . . . . .	1261
<a href="#">constant.cpp</a>	Implementation of <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1262
<a href="#">constant.h</a>	Interface to <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1263
<a href="#">container.h</a>	Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers . . . . .	1264
<a href="#">crc32.h</a>	CRC32 hash function . . . . .	1264
<a href="#">ex.cpp</a>	Implementation of <a href="#">GiNaC</a> 's light-weight expression handles . . . . .	1265

<a href="#">ex.h</a>	Interface to <a href="#">GiNaC</a> 's light-weight expression handles	1265
<a href="#">excompiler.cpp</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1268
<a href="#">excompiler.h</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1268
<a href="#">expair.cpp</a>	Implementation of expression pairs (building blocks of <a href="#">expairseq</a> )	1269
<a href="#">expair.h</a>	Definition of expression pairs (building blocks of <a href="#">expairseq</a> )	1270
<a href="#">expairseq.cpp</a>	Implementation of sequences of expression pairs	1271
<a href="#">expairseq.h</a>	Interface to sequences of expression pairs	1271
<a href="#">exprseq.cpp</a>	Implementation of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1272
<a href="#">exprseq.h</a>	Definition of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1273
<a href="#">factor.cpp</a>	Polynomial factorization (implementation)	1273
<a href="#">factor.h</a>	Polynomial factorization	1275
<a href="#">fail.cpp</a>	Implementation of class signaling failure of operation	1275
<a href="#">fail.h</a>	Interface to class signaling failure of operation	1276
<a href="#">fderivative.cpp</a>	Implementation of abstract derivatives of functions	1276
<a href="#">fderivative.h</a>	Interface to abstract derivatives of functions	1277
<a href="#">flags.h</a>	Collection of all flags used through the <a href="#">GiNaC</a> framework	1277
<a href="#">function.cpp</a>	Implementation of class of symbolic functions	1278
<a href="#">function.h</a>	Interface to class of symbolic functions	1279
<a href="#">ginac.h</a>	This include file includes all other public <a href="#">GiNaC</a> headers	1290
<a href="#">Gt.cpp</a>	Implementation of <a href="#">GiNaC</a> 's elliptic multiple polylogarithms	1291
<a href="#">Gt.h</a>	Interface to <a href="#">GiNaC</a> 's elliptic multiple polylogarithms	1292
<a href="#">Gt_helpers.cpp</a>	Implementation of helper functions for elliptic multiple polylogarithms	1292
<a href="#">Gt_helpers.h</a>	Interface to helper functions for elliptic multiple polylogarithms	1293
<a href="#">hash_map.h</a>	Replacement for <code>map&lt;&gt;</code> using hash tables	1294
<a href="#">hash_seed.h</a>	Type-specific hash seed	1294
<a href="#">idx.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indices	1295
<a href="#">idx.h</a>	Interface to <a href="#">GiNaC</a> 's indices	1296
<a href="#">indexed.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indexed expressions	1297



<a href="#">indexed.h</a>	Interface to <a href="#">GiNaC's indexed expressions</a> . . . . .	1299
<a href="#">inifcns.cpp</a>	Implementation of <a href="#">GiNaC's initially known functions</a> . . . . .	1300
<a href="#">inifcns.h</a>	Interface to <a href="#">GiNaC's initially known functions</a> . . . . .	1303
<a href="#">inifcns_elliptic.cpp</a>	Implementation of some special functions related to elliptic curves . . . . .	1304
<a href="#">inifcns_gamma.cpp</a>	Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff	1306
<a href="#">inifcns_nstdsums.cpp</a>	Implementation of some special functions that have a representation as nested sums . . . . .	1307
<a href="#">inifcns_trans.cpp</a>	Implementation of transcendental (and trigonometric and hyperbolic) functions . . . . .	1309
<a href="#">integral.cpp</a>	Implementation of <a href="#">GiNaC's symbolic integral</a> . . . . .	1312
<a href="#">integral.h</a>	Interface to <a href="#">GiNaC's symbolic integral</a> . . . . .	1313
<a href="#">integration_kernel.cpp</a>	Implementation of <a href="#">GiNaC's integration kernels for iterated integrals</a> . . . . .	1313
<a href="#">integration_kernel.h</a>	Interface to <a href="#">GiNaC's integration kernels for iterated integrals</a> . . . . .	1315
<a href="#">lst.cpp</a>	Implementation of <a href="#">GiNaC's lst</a> . . . . .	1316
<a href="#">lst.h</a>	Definition of <a href="#">GiNaC's lst</a> . . . . .	1317
<a href="#">matrix.cpp</a>	Implementation of symbolic matrices . . . . .	1317
<a href="#">matrix.h</a>	Interface to symbolic matrices . . . . .	1318
<a href="#">mul.cpp</a>	Implementation of <a href="#">GiNaC's products of expressions</a> . . . . .	1320
<a href="#">mul.h</a>	Interface to <a href="#">GiNaC's products of expressions</a> . . . . .	1320
<a href="#">ncmul.cpp</a>	Implementation of <a href="#">GiNaC's non-commutative products of expressions</a> . . . . .	1321
<a href="#">ncmul.h</a>	Interface to <a href="#">GiNaC's non-commutative products of expressions</a> . . . . .	1322
<a href="#">normal.cpp</a>	This file implements several functions that work on univariate and multivariate polynomials and rational functions . . . . .	1323
<a href="#">normal.h</a>	This file defines several functions that work on univariate and multivariate polynomials and rational functions . . . . .	1325
<a href="#">numeric.cpp</a>	This file contains the interface to the underlying bignum package . . . . .	1327
<a href="#">numeric.h</a>	Makes the interface to the underlying bignum package available . . . . .	1330
<a href="#">operators.cpp</a>	Implementation of <a href="#">GiNaC's overloaded operators</a> . . . . .	1333
<a href="#">operators.h</a>	Interface to <a href="#">GiNaC's overloaded operators</a> . . . . .	1335
<a href="#">power.cpp</a>	Implementation of <a href="#">GiNaC's symbolic exponentiation (basis<sup>exponent</sup>)</a> . . . . .	1336
<a href="#">power.h</a>	Interface to <a href="#">GiNaC's symbolic exponentiation (basis<sup>exponent</sup>)</a> . . . . .	1337
<a href="#">print.cpp</a>	Implementation of helper classes for expression output . . . . .	1338

<a href="#">print.h</a>	Definition of helper classes for expression output . . . . .	1338
<a href="#">pseries.cpp</a>	Implementation of class for extended truncated power series and methods for series expansion	1341
<a href="#">pseries.h</a>	Interface to class for extended truncated power series . . . . .	1342
<a href="#">ptr.h</a>	Reference-counted pointer template . . . . .	1342
<a href="#">registrar.cpp</a>	GiNaC's class registrar (for class basic and all classes derived from it) . . . . .	1343
<a href="#">registrar.h</a>	GiNaC's class registrar (for class basic and all classes derived from it) . . . . .	1343
<a href="#">relational.cpp</a>	Implementation of relations between expressions . . . . .	1346
<a href="#">relational.h</a>	Interface to relations between expressions . . . . .	1347
<a href="#">remember.cpp</a>	Implementation of helper classes for using the remember option in GiNaC functions . . . . .	1347
<a href="#">remember.h</a>	Interface to helper classes for using the remember option in GiNaC functions . . . . .	1348
<a href="#">structure.h</a>	Wrapper template for making GiNaC classes out of C++ structures . . . . .	1348
<a href="#">symbol.cpp</a>	Implementation of GiNaC's symbolic objects . . . . .	1349
<a href="#">symbol.h</a>	Interface to GiNaC's symbolic objects . . . . .	1350
<a href="#">symmetry.cpp</a>	Implementation of GiNaC's symmetry definitions . . . . .	1351
<a href="#">symmetry.h</a>	Interface to GiNaC's symmetry definitions . . . . .	1352
<a href="#">tensor.cpp</a>	Implementation of GiNaC's special tensors . . . . .	1353
<a href="#">tensor.h</a>	Interface to GiNaC's special tensors . . . . .	1354
<a href="#">utils.cpp</a>	Implementation of several small and furry utilities needed within GiNaC but not of any interest to the user of the library . . . . .	1355
<a href="#">utils.h</a>	Interface to several small and furry utilities needed within GiNaC but not of any interest to the user of the library . . . . .	1358
<a href="#">utils_multi_iterator.h</a>	Utilities for summing over multiple indices . . . . .	1360
<a href="#">version.h</a>	GiNaC library version information . . . . .	1361
<a href="#">wildcard.cpp</a>	Implementation of GiNaC's wildcard objects . . . . .	1363
<a href="#">wildcard.h</a>	Interface to GiNaC's wildcard objects . . . . .	1364

## Chapter 6

# Directory Documentation

### 6.1 ginac Directory Reference

#### Files

- file [add.cpp](#)  
*Implementation of GiNaC's sums of expressions.*
- file [add.h](#)  
*Interface to GiNaC's sums of expressions.*
- file [archive.cpp](#)  
*Archiving of GiNaC expressions.*
- file [archive.h](#)  
*Archiving of GiNaC expressions.*
- file [assertion.h](#)  
*Assertion macro definition.*
- file [basic.cpp](#)  
*Implementation of GiNaC's ABC.*
- file [basic.h](#)  
*Interface to GiNaC's ABC.*
- file [class\\_info.h](#)  
*Helper templates to provide per-class information for class hierarchies.*
- file [clifford.cpp](#)  
*Implementation of GiNaC's clifford algebra (Dirac gamma) objects.*
- file [clifford.h](#)  
*Interface to GiNaC's clifford algebra (Dirac gamma) objects.*
- file [color.cpp](#)  
*Implementation of GiNaC's color (SU(3) Lie algebra) objects.*
- file [color.h](#)  
*Interface to GiNaC's color (SU(3) Lie algebra) objects.*
- file [compiler.h](#)  
*Definition of optimizing macros.*
- file [constant.cpp](#)  
*Implementation of GiNaC's constant types and some special constants.*
- file [constant.h](#)  
*Interface to GiNaC's constant types and some special constants.*
- file [container.h](#)

- Wrapper template for making GiNaC classes out of STL containers.*
- file [crc32.h](#)  
*CRC32 hash function.*
- file [ex.cpp](#)  
*Implementation of GiNaC's light-weight expression handles.*
- file [ex.h](#)  
*Interface to GiNaC's light-weight expression handles.*
- file [excompiler.cpp](#)  
*Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration.*
- file [excompiler.h](#)  
*Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration.*
- file [expair.cpp](#)  
*Implementation of expression pairs (building blocks of expairseq).*
- file [expair.h](#)  
*Definition of expression pairs (building blocks of expairseq).*
- file [expairseq.cpp](#)  
*Implementation of sequences of expression pairs.*
- file [expairseq.h](#)  
*Interface to sequences of expression pairs.*
- file [exprseq.cpp](#)  
*Implementation of GiNaC's exprseq.*
- file [exprseq.h](#)  
*Definition of GiNaC's exprseq.*
- file [factor.cpp](#)  
*Polynomial factorization (implementation).*
- file [factor.h](#)  
*Polynomial factorization.*
- file [fail.cpp](#)  
*Implementation of class signaling failure of operation.*
- file [fail.h](#)  
*Interface to class signaling failure of operation.*
- file [fderivative.cpp](#)  
*Implementation of abstract derivatives of functions.*
- file [fderivative.h](#)  
*Interface to abstract derivatives of functions.*
- file [flags.h](#)  
*Collection of all flags used through the GiNaC framework.*
- file [function.cpp](#)  
*Implementation of class of symbolic functions.*
- file [function.h](#)  
*Interface to class of symbolic functions.*
- file [ginac.h](#)  
*This include file includes all other public GiNaC headers.*
- file [Gt.cpp](#)  
*Implementation of GiNaC's elliptic multiple polylogarithms.*
- file [Gt.h](#)  
*Interface to GiNaC's elliptic multiple polylogarithms.*
- file [Gt\\_helpers.cpp](#)  
*Implementation of helper functions for elliptic multiple polylogarithms.*
- file [Gt\\_helpers.h](#)  
*Interface to helper functions for elliptic multiple polylogarithms.*

- file [hash\\_map.h](#)  
*Replacement for `map<>` using hash tables.*
- file [hash\\_seed.h](#)  
*Type-specific hash seed.*
- file [idx.cpp](#)  
*Implementation of [GiNaC's](#) indices.*
- file [idx.h](#)  
*Interface to [GiNaC's](#) indices.*
- file [indexed.cpp](#)  
*Implementation of [GiNaC's](#) indexed expressions.*
- file [indexed.h](#)  
*Interface to [GiNaC's](#) indexed expressions.*
- file [inifcns.cpp](#)  
*Implementation of [GiNaC's](#) initially known functions.*
- file [inifcns.h](#)  
*Interface to [GiNaC's](#) initially known functions.*
- file [inifcns\\_elliptic.cpp](#)  
*Implementation of some special functions related to elliptic curves.*
- file [inifcns\\_gamma.cpp](#)  
*Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.*
- file [inifcns\\_nstdsums.cpp](#)  
*Implementation of some special functions that have a representation as nested sums.*
- file [inifcns\\_trans.cpp](#)  
*Implementation of transcendental (and trigonometric and hyperbolic) functions.*
- file [integral.cpp](#)  
*Implementation of [GiNaC's](#) symbolic integral.*
- file [integral.h](#)  
*Interface to [GiNaC's](#) symbolic integral.*
- file [integration\\_kernel.cpp](#)  
*Implementation of [GiNaC's](#) integration kernels for iterated integrals.*
- file [integration\\_kernel.h](#)  
*Interface to [GiNaC's](#) integration kernels for iterated integrals.*
- file [lst.cpp](#)  
*Implementation of [GiNaC's](#) `lst`.*
- file [lst.h](#)  
*Definition of [GiNaC's](#) `lst`.*
- file [matrix.cpp](#)  
*Implementation of symbolic matrices.*
- file [matrix.h](#)  
*Interface to symbolic matrices.*
- file [mul.cpp](#)  
*Implementation of [GiNaC's](#) products of expressions.*
- file [mul.h](#)  
*Interface to [GiNaC's](#) products of expressions.*
- file [ncmul.cpp](#)  
*Implementation of [GiNaC's](#) non-commutative products of expressions.*
- file [ncmul.h](#)  
*Interface to [GiNaC's](#) non-commutative products of expressions.*
- file [normal.cpp](#)  
*This file implements several functions that work on univariate and multivariate polynomials and rational functions.*
- file [normal.h](#)

*This file defines several functions that work on univariate and multivariate polynomials and rational functions.*

- file [numeric.cpp](#)

*This file contains the interface to the underlying bignum package.*

- file [numeric.h](#)

*Makes the interface to the underlying bignum package available.*

- file [operators.cpp](#)

*Implementation of [GiNaC](#)'s overloaded operators.*

- file [operators.h](#)

*Interface to [GiNaC](#)'s overloaded operators.*

- file [power.cpp](#)

*Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).*

- file [power.h](#)

*Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).*

- file [print.cpp](#)

*Implementation of helper classes for expression output.*

- file [print.h](#)

*Definition of helper classes for expression output.*

- file [pseries.cpp](#)

*Implementation of class for extended truncated power series and methods for series expansion.*

- file [pseries.h](#)

*Interface to class for extended truncated power series.*

- file [ptr.h](#)

*Reference-counted pointer template.*

- file [registrar.cpp](#)

*[GiNaC](#)'s class registrar (for class basic and all classes derived from it).*

- file [registrar.h](#)

*[GiNaC](#)'s class registrar (for class basic and all classes derived from it).*

- file [relational.cpp](#)

*Implementation of relations between expressions.*

- file [relational.h](#)

*Interface to relations between expressions.*

- file [remember.cpp](#)

*Implementation of helper classes for using the remember option in [GiNaC](#) functions.*

- file [remember.h](#)

*Interface to helper classes for using the remember option in [GiNaC](#) functions.*

- file [structure.h](#)

*Wrapper template for making [GiNaC](#) classes out of C++ structures.*

- file [symbol.cpp](#)

*Implementation of [GiNaC](#)'s symbolic objects.*

- file [symbol.h](#)

*Interface to [GiNaC](#)'s symbolic objects.*

- file [symmetry.cpp](#)

*Implementation of [GiNaC](#)'s symmetry definitions.*

- file [symmetry.h](#)

*Interface to [GiNaC](#)'s symmetry definitions.*

- file [tensor.cpp](#)

*Implementation of [GiNaC](#)'s special tensors.*

- file [tensor.h](#)

*Interface to [GiNaC](#)'s special tensors.*

- file [utils.cpp](#)

*Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.*

- file [utils.h](#)  
*Interface to several small and furry utilities needed within GiNaC but not of any interest to the user of the library.*
- file [utils\\_multi\\_iterator.h](#)  
*Utilities for summing over multiple indices.*
- file [version.h](#)  
*GiNaC library version information.*
- file [wildcard.cpp](#)  
*Implementation of GiNaC's wildcard objects.*
- file [wildcard.h](#)  
*Interface to GiNaC's wildcard objects.*





# Chapter 7

## Namespace Documentation

### 7.1 GiNaC Namespace Reference

#### Namespaces

- namespace [internal](#)
- namespace [Gt\\_detail](#)

#### Classes

- class [add](#)  
*Sum of expressions.*
- class [archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).*
- class [unarchive\\_table\\_t](#)
- class [archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).*
- struct [evalf\\_map\\_function](#)  
*Function object to be applied by [basic::evalf\(\)](#).*
- struct [evalm\\_map\\_function](#)  
*Function object to be applied by [basic::evalm\(\)](#).*
- struct [eval\\_integ\\_map\\_function](#)  
*Function object to be applied by [basic::eval\\_integ\(\)](#).*
- struct [derivative\\_map\\_function](#)  
*Function object to be applied by [basic::derivative\(\)](#).*
- struct [expand\\_map\\_function](#)  
*Function object to be applied by [basic::expand\(\)](#).*
- struct [map\\_function](#)  
*Function object for [map\(\)](#).*
- class [visitor](#)  
*Degenerate base class for visitors.*
- class [basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*
- class [class\\_info](#)
- struct [is\\_not\\_a\\_clifford](#)

- Predicate for finding non-clifford objects.*

  - class [clifford](#)

*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
  - class [diracone](#)

*This class represents the Clifford algebra unity element.*
  - class [cliffordunit](#)

*This class represents the Clifford algebra generators (units).*
  - class [diracgamma](#)

*This class represents the Dirac gamma Lorentz vector.*
  - class [diracgamma5](#)

*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
  - class [diracgammaL](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1-\text{gamma}5)$ .*
  - class [diracgammaR](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1+\text{gamma}5)$ .*
  - class [color](#)

*This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.*
  - class [su3one](#)

*This class represents the  $su(3)$  unity element.*
  - class [su3t](#)

*This class represents an  $su(3)$  generator.*
  - class [su3f](#)

*This class represents the tensor of antisymmetric  $su(3)$  structure constants.*
  - class [su3d](#)

*This class represents the tensor of symmetric  $su(3)$  structure constants.*
  - class [constant](#)

*This class holds constants, symbols with specific numerical value.*
  - class [container\\_storage](#)

*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
  - class [container](#)

*Wrapper template for making [GiNaC](#) classes out of STL containers.*
  - class [library\\_init](#)

*Helper class to initialize the library.*
  - class [ex](#)

*Lightweight wrapper for [GiNaC](#)'s symbolic objects.*
  - class [const\\_iterator](#)
  - class [const\\_preorder\\_iterator](#)
  - class [const\\_postorder\\_iterator](#)
  - struct [ex\\_is\\_less](#)
  - struct [ex\\_is\\_equal](#)
  - struct [op0\\_is\\_equal](#)
  - struct [ex\\_swap](#)
  - class [pointer\\_to\\_map\\_function](#)
  - class [pointer\\_to\\_map\\_function\\_1arg](#)
  - class [pointer\\_to\\_map\\_function\\_2args](#)
  - class [pointer\\_to\\_map\\_function\\_3args](#)
  - class [pointer\\_to\\_member\\_to\\_map\\_function](#)
  - class [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#)
  - class [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#)
  - class [pointer\\_to\\_member\\_to\\_map\\_function\\_3args](#)
  - class [expair](#)

- A pair of expressions.*
- struct [expair\\_is\\_less](#)
  - Function object for insertion into third argument of STL's sort() etc.*
- struct [expair\\_rest\\_is\\_less](#)
  - Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().*
- struct [expair\\_swap](#)
- class [expairseq](#)
  - A sequence of class expair.*
- class [make\\_flat\\_inserter](#)
  - Class to handle the renaming of dummy indices.*
- class [fail](#)
- class [fderivative](#)
  - This class represents the (abstract) derivative of a symbolic function.*
- class [expand\\_options](#)
  - Flags to control the behavior of [expand\(\)](#).*
- class [has\\_options](#)
  - Flags to control the behavior of [has\(\)](#).*
- class [subs\\_options](#)
  - Flags to control the behavior of [subs\(\)](#).*
- class [domain](#)
  - Domain of an object.*
- class [series\\_options](#)
  - Flags to control series expansion.*
- class [determinant\\_algo](#)
  - Switch to control algorithm for determinant computation.*
- class [solve\\_algo](#)
  - Switch to control algorithm for linear system solving.*
- class [status\\_flags](#)
  - Flags to store information about the state of an object.*
- class [info\\_flags](#)
  - Possible attributes an object can have.*
- class [return\\_types](#)
- class [remember\\_strategies](#)
  - Strategies how to clean up the function remember cache.*
- class [factor\\_options](#)
  - Flags to control the polynomial factorization.*
- class [function\\_options](#)
- class [do\\_taylor](#)
  - Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*
- class [function](#)
  - The class function is used to implement builtin functions like sin, cos... and user defined functions.*
- class [Gt](#)
  - Elliptic Multiple Polylogarithm.*
- class [idx](#)
  - This class holds one index of an indexed object.*
- class [varidx](#)
  - This class holds an index with a variance (co- or contravariant).*
- class [spinidx](#)
  - This class holds a spinor index that can be dotted or undotted and that also has a variance.*
- struct [idx\\_is\\_equal\\_ignore\\_dim](#)

- struct [is\\_summation\\_idx](#)
- struct [ex\\_base\\_is\\_less](#)
- class [terminfo](#)

*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*

- class [terminfo\\_is\\_less](#)
- class [symminfo](#)

*This structure stores the individual symmetrized terms obtained during the simplification of sums.*

- class [symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [symminfo\\_is\\_less\\_by\\_orig](#)
- class [indexed](#)

*This class holds an indexed expression.*

- class [smapkey](#)
- class [scalar\\_products](#)

*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

- class [symbolset](#)
- class [zeta1\\_SERIAL](#)

*Complex conjugate.*

- class [zeta2\\_SERIAL](#)

*Alternating Euler sum or colored MZV.*

- class [G2\\_SERIAL](#)

*Generalized multiple polylogarithm.*

- class [G3\\_SERIAL](#)

*Generalized multiple polylogarithm with explicit imaginary parts.*

- class [psi1\\_SERIAL](#)

*Polylogarithm and multiple polylogarithm.*

- class [psi2\\_SERIAL](#)

*Derivatives of Psi-function (aka polygamma-functions).*

- class [iterated\\_integral2\\_SERIAL](#)

*Complete elliptic integral of the first kind.*

- class [iterated\\_integral3\\_SERIAL](#)

*Iterated integral with explicit truncation.*

- struct [error\\_and\\_integral](#)
- struct [error\\_and\\_integral\\_is\\_less](#)
- class [integral](#)

*Symbolic integral.*

- class [integration\\_kernel](#)

*The base class for integration kernels for iterated integrals.*

- class [basic\\_log\\_kernel](#)

*The basic integration kernel with a logarithmic singularity at the origin.*

- class [multiple\\_polylog\\_kernel](#)

*The integration kernel for multiple polylogarithms.*

- class [ELi\\_kernel](#)

*The ELi-kernel.*

- class [Ebar\\_kernel](#)

*The Ebar-kernel.*

- class [Kronecker\\_dtau\\_kernel](#)

*The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).*

- class [Kronecker\\_dz\\_kernel](#)

*The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .*

- class [Eisenstein\\_kernel](#)

- The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

  - class [Eisenstein\\_h\\_kernel](#)
- The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

  - class [modular\\_form\\_kernel](#)
- A kernel corresponding to a polynomial in Eisenstein series.

  - class [user\\_defined\\_kernel](#)
- A user-defined integration kernel.

  - class [matrix](#)
- Symbolic matrices.

  - class [mul](#)
- Product of expressions.

  - class [ncmul](#)
- Non-commutative product of expressions.

  - struct [sym\\_desc](#)
- This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

  - class [gcdheu\\_failed](#)
- Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

  - struct [normal\\_map\\_function](#)
- Function object to be applied by [basic::normal\(\)](#).

  - struct [gcd\\_options](#)
- Flags to control the behavior of [gcd\(\)](#) and friends.

  - class [lanczos\\_coeffs](#)
  - class [\\_numeric\\_digits](#)
- This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

  - class [pole\\_error](#)
- Exception class thrown when a singularity is encountered.

  - class [numeric](#)
- This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

  - class [power](#)
- This class holds a two-component object, a basis and an exponent representing exponentiation.

  - class [print\\_context\\_options](#)
- This class stores information about a registered [print\\_context](#) class.

  - class [print\\_options](#)
- Flags to control the behavior of a [print\\_context](#).

  - class [print\\_context](#)
- Base class for [print\\_contexts](#).

  - class [print\\_dflt](#)
- Context for default (ginsh-parsable) output.

  - class [print\\_latex](#)
- Context for latex-parsable output.

  - class [print\\_python](#)
- Context for python pretty-print output.

  - class [print\\_python\\_repr](#)
- Context for python-parsable output.

  - class [print\\_tree](#)
- Context for tree-like output for debugging.

  - class [print\\_csrc](#)
- Base context for C source output.

  - class [print\\_csrc\\_float](#)
- Context for C source output using float precision.

- class [print\\_csrc\\_double](#)  
*Context for C source output using double precision.*
- class [print\\_csrc\\_cl\\_N](#)  
*Context for C source output using CLN numbers.*
- class [print\\_functor\\_impl](#)  
*Base class for [print\\_functor](#) handlers.*
- class [print\\_ptrfun\\_handler](#)  
*[print\\_functor](#) handler for pointer-to-functions of class *T*, context type *C**
- class [print\\_memfun\\_handler](#)  
*[print\\_functor](#) handler for member functions of class *T*, context type *C**
- class [print\\_functor](#)  
*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*
- class [pseries](#)  
*This class holds a extended truncated power series (positive and negative integer powers).*
- class [refcounted](#)  
*Base class for reference-counted objects.*
- class [ptr](#)  
*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
- struct [return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [registered\\_class\\_options](#)  
*This class stores information about a registered [GiNaC](#) class.*
- class [relational](#)  
*This class holds a relation consisting of two expressions and a logical relation between them.*
- class [remember\\_table\\_entry](#)  
*A single entry in the remember table of a function.*
- class [remember\\_table\\_list](#)  
*A list of entries in the remember table having some least significant bits of the hashvalue in common.*
- class [remember\\_table](#)  
*The remember table is organized like an n-fold associative cache in a microprocessor.*
- class [compare\\_all\\_equal](#)  
*Comparison policy: all structures of one type are equal.*
- class [compare\\_std\\_less](#)  
*Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.*
- class [compare\\_bitwise](#)  
*Comparison policy: use bit-wise comparison to compare structures.*
- class [structure](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*
- class [symbol](#)  
*Basic CAS symbol.*
- class [realsymbol](#)  
*Specialization of symbol to real domain.*
- class [possymbol](#)  
*Specialization of symbol to real positive domain.*
- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- class [symmetry](#)  
*This class describes the symmetry of a group of indices.*
- class [tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*
- class [tensdelta](#)

- This class represents the delta tensor.*

  - class [tensmetric](#)
- This class represents a general metric tensor which can be used to raise/lower indices.*

  - class [minkmetric](#)
- This class represents a Minkowski metric tensor.*

  - class [spinmetric](#)
- This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*

  - class [tensepsilon](#)
- This class represents the totally antisymmetric epsilon tensor.*

  - class [dunno](#)
- Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold().*

  - class [basic\\_partition\\_generator](#)
- Base class for generating all bounded combinatorial partitions of an integer n with exactly m parts in non-decreasing order.*

  - class [partition\\_with\\_zero\\_parts\\_generator](#)
- Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order.*

  - class [partition\\_generator](#)
- Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order.*

  - class [composition\\_generator](#)
- Generate all compositions of a partition of an integer n, starting with the compositions which has non-decreasing order.*

  - class [has\\_distance](#)
- SFINAE test for distance.*

  - class [basic\\_multi\\_iterator](#)
- [basic\\_multi\\_iterator](#) is a base class.*

  - class [multi\\_iterator\\_ordered](#)
- The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*

  - class [multi\\_iterator\\_ordered\\_eq](#)
- The class [multi\\_iterator\\_ordered\\_eq](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*

  - class [multi\\_iterator\\_ordered\\_eq\\_indv](#)
- The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*

  - class [multi\\_iterator\\_counter](#)
- The class [multi\\_iterator\\_counter](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*

  - class [multi\\_iterator\\_counter\\_indv](#)
- The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*

  - class [multi\\_iterator\\_permutation](#)
- The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.*

  - class [multi\\_iterator\\_shuffle](#)
- The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.*

  - class [multi\\_iterator\\_shuffle\\_prime](#)
- The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).*

  - class [wildcard](#)
- This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*

## Typedefs

- typedef unsigned [archive\\_node\\_id](#)  
Numerical ID value to refer to an [archive\\_node](#).
- typedef unsigned [archive\\_atom](#)  
Numerical ID value to refer to a string.
- typedef [basic](#) [\\*\(\\* synthesize\\_func\)](#) ()
- typedef std::map< std::string, [synthesize\\_func](#) > [unarchive\\_map\\_t](#)
- typedef std::vector< [ex](#) > [exvector](#)
- typedef std::set< [ex](#), [ex\\_is\\_less](#) > [exset](#)
- typedef std::map< [ex](#), [ex](#), [ex\\_is\\_less](#) > [exmap](#)
- typedef [ex](#) [\\* evalfunctype](#) ()
- typedef double [\\* FUNCP\\_1P](#) (double)  
Function pointer with one function parameter.
- typedef double [\\* FUNCP\\_2P](#) (double, double)  
Function pointer with two function parameters.
- typedef void [\\* FUNCP\\_CUBA](#) (const int \*, const double[], const int \*, double[])  
Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).
- typedef std::vector< [expair](#) > [epvector](#)  
*expair-vector*
- typedef [epvector::iterator](#) [epp](#)  
*expair-vector pointer*
- typedef [container](#)< std::vector > [exprseq](#)
- typedef std::multiset< unsigned > [paramset](#)
- typedef [ex](#) [\\* eval\\_funcp](#) ()
- typedef [ex](#) [\\* evalf\\_funcp](#) ()
- typedef [ex](#) [\\* conjugate\\_funcp](#) ()
- typedef [ex](#) [\\* real\\_part\\_funcp](#) ()
- typedef [ex](#) [\\* imag\\_part\\_funcp](#) ()
- typedef [ex](#) [\\* expand\\_funcp](#) ()
- typedef [ex](#) [\\* derivative\\_funcp](#) ()
- typedef [ex](#) [\\* expl\\_derivative\\_funcp](#) ()
- typedef [ex](#) [\\* power\\_funcp](#) ()
- typedef [ex](#) [\\* series\\_funcp](#) ()
- typedef void [\\* print\\_funcp](#) ()
- typedef bool [\\* info\\_funcp](#) ()
- typedef [ex](#) [\\* eval\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* evalf\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* conjugate\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* real\\_part\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* imag\\_part\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* expand\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* derivative\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* expl\\_derivative\\_funcp\\_1](#) (const [ex](#) &, const [symbol](#) &)
- typedef [ex](#) [\\* power\\_funcp\\_1](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* series\\_funcp\\_1](#) (const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void [\\* print\\_funcp\\_1](#) (const [ex](#) &, const [print\\_context](#) &)
- typedef bool [\\* info\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* eval\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* evalf\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* conjugate\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* real\\_part\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* imag\\_part\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* expand\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &, unsigned)



- [illegible]



- [illegible]



- typedef [ex](#)(\* [power\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_exvector](#)) (const [exvector](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_exvector](#)) (const [exvector](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_exvector](#)) (const [exvector](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- template<typename T, class Hash = std::hash<[ex](#)>, class KeyEqual = std::equal\_to<[ex](#)>, class Allocator = std::allocator<std::pair<const [ex](#), T>>>
 using [exhashmap](#) = std::unordered\_map<[ex](#), T, Hash, KeyEqual, Allocator>
- typedef std::map< [spmapkey](#), [ex](#) > [spmap](#)
- typedef map< [error\\_and\\_integral](#), [ex](#), [error\\_and\\_integral\\_is\\_less](#) > [lookup\\_map](#)
- typedef [container](#)< std::list > [lst](#)
- typedef std::vector< std::size\_t > [uintvector](#)
- typedef std::vector< unsigned > [unsignedvector](#)
- typedef std::vector< [exvector](#) > [exvectorvector](#)

- typedef std::vector< [sym\\_desc](#) > [sym\\_desc\\_vec](#)
- typedef void(\* [digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*
- typedef [class\\_info](#)< [print\\_context\\_options](#) > [print\\_context\\_class\\_info](#)
- typedef [class\\_info](#)< [registered\\_class\\_options](#) > [registered\\_class\\_info](#)

## Enumerations

- enum { [callback\\_registered](#) = 1 }

## Functions

- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([add](#), [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&[add::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[add::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_csrc](#) >(&[add::do\\_print\\_csrc](#)), [print\\_func](#)< [print\\_tree](#) >(&[add::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[add::do\\_print\\_python\\_repr](#)))  
[add](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([add](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([add](#))
- static void [write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive\\_node](#) &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive](#) &r)  
*Write archive to binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive\\_node](#) &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive](#) &r)  
*Read archive from binary data stream.*
- static [synthesize\\_func](#) [find\\_factory\\_fcn](#) (const std::string &name)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([basic](#), void, [print\\_func](#)< [print\\_context](#) >(&[basic::do\\_print](#)), [print\\_func](#)< [print\\_tree](#) >(&[basic::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[basic::do\\_print\\_python\\_repr](#)))  
[basic](#)  
*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.*
- template<class T>  
bool [is\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, including base classes.*
- template<class T>  
bool [is\\_exactly\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, not including base classes.*
- template<class B, typename... Args>  
B & [dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B>  
B & [dynallocate](#) (std::initializer\_list< [ex](#) > il)  
*Constructs a new (class basic or derived) B object on the heap.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([clifford](#), [indexed](#), [print\\_func](#)< [print\\_dflt](#) >(&[clifford::do\\_print\\_dflt](#)), [print\\_func](#)< [print\\_latex](#) >(&[clifford::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[clifford::do\\_print\\_tree](#)))  
[GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([diracone](#)
- [print\\_func](#)< [print\\_dflt](#) > (&[diracone::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[diracone](#)



- [GINAC\\_BIND\\_UNARCHIVER](#) ([clifford](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([cliffordunit](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracone](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma5](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaL](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaR](#))
- static bool [is\\_dirac\\_slash](#) (const [ex](#) &seq0)
- static void [base\\_and\\_index](#) (const [ex](#) &c, [ex](#) &b, [ex](#) &i)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$ .*
- [ex](#) [dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned [get\\_dim\\_uint](#) (const [ex](#) &e)
- [ex](#) [clifford\\_unit](#) (const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- [ex](#) [dirac\\_gamma](#) (const [ex](#) &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- [ex](#) [dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- [ex](#) [dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- [ex](#) [dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- [ex](#) [dirac\\_slash](#) (const [ex](#) &e, const [ex](#) &dim, unsigned char rl=0)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
- static unsigned char [get\\_representation\\_label](#) (const [return\\_type\\_t](#) &ti)  
*Extract representation label from tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)).*
- static [ex](#) [trace\\_string](#) (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, const std::set< unsigned char > &rls, const [ex](#) &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, const [lst](#) &rls, const [ex](#) &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, unsigned char rl=0, const [ex](#) &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- [ex](#) [canonicalize\\_clifford](#) (const [ex](#) &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- [ex](#) [clifford\\_star\\_bar](#) (const [ex](#) &e, bool do\_bar, unsigned options)  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- [ex](#) [clifford\\_prime](#) (const [ex](#) &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- [ex](#) [remove\\_dirac\\_ONE](#) (const [ex](#) &e, unsigned char rl=0, unsigned options=0)  
*Replaces [dirac\\_ONE](#)'s (with a [representation\\_label](#) no less than  $rl$ ) in  $e$  with 1.*
- int [clifford\\_max\\_label](#) (const [ex](#) &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if  $e$  contains at least one, otherwise returns -1.*
- [ex](#) [clifford\\_norm](#) (const [ex](#) &e)  
*Calculation of the norm in the Clifford algebra.*
- [ex](#) [clifford\\_inverse](#) (const [ex](#) &e)  
*Calculation of the inverse in the Clifford algebra.*
- [ex](#) [lst\\_to\\_clifford](#) (const [ex](#) &v, const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)  
*List or vector conversion into the Clifford vector.*

- `ex lst_to_clifford` (const `ex` &v, const `ex` &e)  
*List or vector conversion into the Clifford vector.*
- static `ex get_clifford_comp` (const `ex` &e, const `ex` &c, bool root=true)  
*Auxiliary structure to define a function for stripping one Clifford unit from vectors.*
- `lst clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)  
*An inverse function to `lst_to_clifford()`.*
- `ex clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*
- `ex clifford_moebius_map` (const `ex` &M, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*
- `GINAC_DECLARE_UNARCHIVER` (clifford)
- `GINAC_DECLARE_UNARCHIVER` (diracone)
- `GINAC_DECLARE_UNARCHIVER` (cliffordunit)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma5)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaL)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaR)
- bool `is_clifford_tinfo` (const `return_type_t` &ti)  
*Check whether a given `return_type_t` object (as returned by `return_type_tinfo()`) is that of a clifford object (with an arbitrary representation label).*
- `ex clifford_bar` (const `ex` &e)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- `ex clifford_star` (const `ex` &e)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (su3one, tensor, print\_func< `print_dflt` >(&su3one::do\_print). print\_func< `print_latex` >(&su3one::do\_print\_latex)) `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT`(su3t  
`print_func< print_dflt > (&su3t::do_print). print_func< print_latex >(&su3t`  
`GINAC_BIND_UNARCHIVER` (color)  
`GINAC_BIND_UNARCHIVER` (su3one)  
`GINAC_BIND_UNARCHIVER` (su3t)  
`GINAC_BIND_UNARCHIVER` (su3f)  
`GINAC_BIND_UNARCHIVER` (su3d)
- static `ex permute_free_index_to_front` (const `exvector` &iv3, const `exvector` &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- `ex color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- static bool `is_color_tinfo` (const `return_type_t` &ti)  
*Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).*
- static unsigned char `get_representation_label` (const `return_type_t` &ti)



- Extract representation label from tinfo key (as returned by return\_type\_tinfo()).*
- `ex color_trace` (const `ex` &`e`, const `std::set< unsigned char >` &`rls`)
- Calculate color traces over the specified set of representation labels.*
- `ex color_trace` (const `ex` &`e`, const `lst` &`rl`)
- Calculate color traces over the specified list of representation labels.*
- `ex color_trace` (const `ex` &`e`, unsigned char `rl=0`)
- Calculate the trace of an expression containing color objects with a specified representation label.*
- `GINAC_DECLARE_UNARCHIVER` (`color`)
- `GINAC_DECLARE_UNARCHIVER` (`su3one`)
- `GINAC_DECLARE_UNARCHIVER` (`su3t`)
- `GINAC_DECLARE_UNARCHIVER` (`su3f`)
- `GINAC_DECLARE_UNARCHIVER` (`su3d`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`constant`, `basic`, `print_func< print_context >` (&`constant::do_print`), `print_func< print_latex >` (&`constant::do_print_latex`), `print_func< print_tree >` (&`constant::do_print_tree`), `print_func< print_python_repr >` (&`constant::do_print_python_repr`)) const ant
- `GINAC_BIND_UNARCHIVER` (`constant`)
- `GINAC_DECLARE_UNARCHIVER` (`constant`)
- static unsigned `crc32` (const char \*`c`, const unsigned `len`, const unsigned `crcinit`)
- bool `are_ex_trivially_equal` (const `ex` &`e1`, const `ex` &`e2`)
- Compare two objects of class quickly without doing a deep tree traversal.*
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exvector` &`e`)
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exset` &`e`)
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exmap` &`e`)
- `size_t nops` (const `ex` &`thisex`)
- `ex expand` (const `ex` &`thisex`, unsigned `options=0`)
- `ex conjugate` (const `ex` &`thisex`)
- `ex real_part` (const `ex` &`thisex`)
- `ex imag_part` (const `ex` &`thisex`)
- bool `has` (const `ex` &`thisex`, const `ex` &`pattern`, unsigned `options=0`)
- bool `find` (const `ex` &`thisex`, const `ex` &`pattern`, `exset` &`found`)
- bool `is_polynomial` (const `ex` &`thisex`, const `ex` &`vars`)
- int `degree` (const `ex` &`thisex`, const `ex` &`s`)
- int `ldegree` (const `ex` &`thisex`, const `ex` &`s`)
- `ex coeff` (const `ex` &`thisex`, const `ex` &`s`, int `n=1`)
- `ex numer` (const `ex` &`thisex`)
- `ex denom` (const `ex` &`thisex`)
- `ex numer_denom` (const `ex` &`thisex`)
- `ex normal` (const `ex` &`thisex`)
- `ex to_rational` (const `ex` &`thisex`, `exmap` &`repl`)
- `ex to_polynomial` (const `ex` &`thisex`, `exmap` &`repl`)
- `ex collect` (const `ex` &`thisex`, const `ex` &`s`, bool `distributed=false`)
- `ex eval` (const `ex` &`thisex`)
- `ex evalf` (const `ex` &`thisex`)
- `ex evalm` (const `ex` &`thisex`)
- `ex eval_integ` (const `ex` &`thisex`)
- `ex diff` (const `ex` &`thisex`, const `symbol` &`s`, unsigned `nth=1`)
- `ex series` (const `ex` &`thisex`, const `ex` &`r`, int `order`, unsigned `options=0`)
- bool `match` (const `ex` &`thisex`, const `ex` &`pattern`, `exmap` &`repl_lst`)
- `ex simplify_indexed` (const `ex` &`thisex`, unsigned `options=0`)
- `ex simplify_indexed` (const `ex` &`thisex`, const `scalar_products` &`sp`, unsigned `options=0`)
- `ex symmetrize` (const `ex` &`thisex`)
- `ex symmetrize` (const `ex` &`thisex`, const `lst` &`l`)
- `ex antisymmetrize` (const `ex` &`thisex`)
- `ex antisymmetrize` (const `ex` &`thisex`, const `lst` &`l`)

- [ex symmetrize\\_cyclic](#) (const [ex](#) &thisex)
- [ex symmetrize\\_cyclic](#) (const [ex](#) &thisex, const [lst](#) &l)
- [ex op](#) (const [ex](#) &thisex, size\_t i)
- [ex lhs](#) (const [ex](#) &thisex)
- [ex rhs](#) (const [ex](#) &thisex)
- bool [is\\_zero](#) (const [ex](#) &thisex)
- void [swap](#) ([ex](#) &e1, [ex](#) &e2)
- [ex subs](#) (const [ex](#) &thisex, const [exmap](#) &m, unsigned options=0)
- [ex subs](#) (const [ex](#) &thisex, const [lst](#) &ls, const [lst](#) &lr, unsigned options=0)
- [ex subs](#) (const [ex](#) &thisex, const [ex](#) &e, unsigned options=0)
- template<class T>  
bool [is\\_a](#) (const [ex](#) &obj)  
*Check if ex is a handle to a T, including base classes.*
- template<class T>  
bool [is\\_exactly\\_a](#) (const [ex](#) &obj)  
*Check if ex is a handle to a T, not including base classes.*
- template<class T>  
[attribute\\_pure](#) const T & [ex\\_to](#) (const [ex](#) &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- void [compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym, [FUNCP\\_1P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym1, const [symbol](#) &sym2, [FUNCP\\_2P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [compile\\_ex](#) (const [lst](#) &exprs, const [lst](#) &syms, [FUNCP\\_CUBA](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [link\\_ex](#) (const std::string filename, [FUNCP\\_1P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_1P](#) to the contained function.*
- void [link\\_ex](#) (const std::string filename, [FUNCP\\_2P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_2P](#) to the contained function.*
- void [link\\_ex](#) (const std::string filename, [FUNCP\\_CUBA](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_CUBA](#) to the contained function.*
- void [unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*
- void [swap](#) ([expair](#) &e1, [expair](#) &e2)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([expairseq](#), [basic](#), print\_func< [print\\_context](#) >(&[expairseq](#)::do\_print). print\_func< [print\\_tree](#) >(&[expairseq](#)::do\_print\_tree)) class [exp\\_is\\_less](#)
- [epvector](#) \* [conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*
- [ex factor](#) (const [ex](#) &poly, unsigned options)  
*Interface function to the outside world.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fail](#), [basic](#), print\_func< [print\\_context](#) >(&[fail](#)::do\_print). print\_func< [print\\_tree](#) >(&[fail](#)::do\_print\_tree)) [GINAC\\_BIND\\_UNARCHIVER](#)([fail](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([fail](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fderivative](#), [function](#), print\_func< [print\\_context](#) >(&[fderivative](#)::do\_print). print\_func< [print\\_latex](#) >(&[fderivative](#)::do\_print\_latex). print\_func< [print\\_csrc](#) >(&[fderivative](#)::do\_print\_csrc). print\_func< [print\\_tree](#) >(&[fderivative](#)::do\_print\_tree)) [fderivative](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([fderivative](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([fderivative](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([function](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([function](#))
- template<typename T>  
bool [is\\_the\\_function](#) (const [ex](#) &x)

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (Gt, basic, print\_func< print\_dflt >(&Gt::do\_print). print\_func< print\_python >(&Gt::do\_print). print\_func< print\_latex >(&Gt::do\_print\_latex)) Gt
- `GINAC_BIND_UNARCHIVER` (Gt)
- `GINAC_DECLARE_UNARCHIVER` (Gt)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (idx, basic, print\_func< print\_context >(&idx::do\_print). print\_func< print\_latex >(&idx::do\_print\_latex). print\_func< print\_csrc >(&idx::do\_print\_csrc). print\_func< print\_tree >(&idx::do\_print\_tree)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(varidx
- `print_func< print_context >` (&varidx::do\_print). print\_func< print\_latex >(&varidx
- `GINAC_BIND_UNARCHIVER` (idx)
- `GINAC_BIND_UNARCHIVER` (varidx)
- `GINAC_BIND_UNARCHIVER` (spinidx)
- `bool is_dummy_pair` (const idx &i1, const idx &i2)  
*Check whether two indices form a dummy pair.*
- `bool is_dummy_pair` (const ex &e1, const ex &e2)  
*Check whether two expressions form a dummy index pair.*
- `void find_free_and_dummy` (exvector::const\_iterator it, exvector::const\_iterator itend, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- `ex minimal_dim` (const ex &dim1, const ex &dim2)  
*Return the minimum of two index dimensions.*
- `GINAC_DECLARE_UNARCHIVER` (idx)
- `GINAC_DECLARE_UNARCHIVER` (varidx)
- `GINAC_DECLARE_UNARCHIVER` (spinidx)
- `void find_free_and_dummy` (const exvector &v, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- `void find_dummy_indices` (const exvector &v, exvector &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- `size_t count_dummy_indices` (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- `size_t count_free_indices` (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (indexed, exprseq, print\_func< print\_context >(&indexed::do\_print). print\_func< print\_latex >(&indexed::do\_print\_latex). print\_func< print\_tree >(&indexed::do\_print\_tree)) indexed
- `GINAC_BIND_UNARCHIVER` (indexed)
- `static bool indices_consistent` (const exvector &v1, const exvector &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- `template<class T>`  
`size_t number_of_type` (const exvector &v)
- `template<class T>`  
`static ex rename_dummy_indices` (const ex &e, exvector &global\_dummy\_indices, exvector &local\_dummy←  
\_indices)  
*Rename dummy indices in an expression.*
- `static void find_variant_indices` (const exvector &v, exvector &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- `bool reposition_dummy_indices` (ex &e, exvector &variant\_dummy\_indices, exvector &moved\_indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- `static void product_to_exvector` (const ex &e, exvector &v, bool &non\_commutative)
- `template<class T>`  
`ex idx_symmetrization` (const ex &r, const exvector &local\_dummy\_indices)
- `ex simplify_indexed` (const ex &e, exvector &free\_indices, exvector &dummy\_indices, const scalar\_products &sp)

*Simplify indexed expression, return list of free indices.*

- `ex simplify_indexed_product` (const `ex` &e, `exvector` &free\_indices, `exvector` &dummy\_indices, const `scalar_products` &sp)

*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*

- `bool hasindex` (const `ex` &x, const `ex` &sym)
- `exvector get_all_dummy_indices_safely` (const `ex` &e)

*More reliable version of the form.*

- `exvector get_all_dummy_indices` (const `ex` &e)

*Returns all dummy indices from the exvector.*

- `lst rename_dummy_indices_uniquely` (const `exvector` &va, const `exvector` &vb)

*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*

- `ex rename_dummy_indices_uniquely` (const `exvector` &va, const `exvector` &vb, const `ex` &b)

*Same as above, where va and vb contain the indices of a and b and are sorted.*

- `ex rename_dummy_indices_uniquely` (const `ex` &a, const `ex` &b)

*Returns b with all dummy indices, which are common with a, renamed.*

- `ex rename_dummy_indices_uniquely` (`exvector` &va, const `ex` &b, bool modify\_va=false)

*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*

- `ex expand_dummy_sum` (const `ex` &e, bool subs\_idx=false)

*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

- `GINAC_DECLARE_UNARCHIVER` (indexed)
- static `ex conjugate_evalf` (const `ex` &arg)
- static `ex conjugate_eval` (const `ex` &arg)
- static void `conjugate_print_latex` (const `ex` &arg, const `print_context` &c)
- static `ex conjugate_conjugate` (const `ex` &arg)
- static `ex conjugate_expl_derivative` (const `ex` &arg, const `symbol` &s)
- static `ex conjugate_real_part` (const `ex` &arg)
- static `ex conjugate_imag_part` (const `ex` &arg)
- static bool `func_arg_info` (const `ex` &arg, unsigned inf)
- static bool `conjugate_info` (const `ex` &arg, unsigned inf)
- `REGISTER_FUNCTION` (conjugate\_function, eval\_func(conjugate\_eval). evalf\_func(conjugate\_evalf). expl\_derivative\_func(conjugate\_expl\_derivative). info\_func(conjugate\_info). print\_func< `print_latex` >(conjugate\_print\_latex). conjugate\_func(conjugate\_conjugate). real\_part\_func(conjugate\_real\_part). imag\_part\_func(conjugate\_imag\_part). set\_name("conjugate","conjugate"))
- static `ex real_part_evalf` (const `ex` &arg)
- static `ex real_part_eval` (const `ex` &arg)
- static void `real_part_print_latex` (const `ex` &arg, const `print_context` &c)
- static `ex real_part_conjugate` (const `ex` &arg)
- static `ex real_part_real_part` (const `ex` &arg)
- static `ex real_part_imag_part` (const `ex` &arg)
- static `ex real_part_expl_derivative` (const `ex` &arg, const `symbol` &s)
- `REGISTER_FUNCTION` (real\_part\_function, eval\_func(real\_part\_eval). evalf\_func(real\_part\_evalf). expl\_derivative\_func(real\_part\_expl\_derivative). print\_func< `print_latex` >(real\_part\_print\_latex). conjugate\_func(real\_part\_conjugate). real\_part\_func(real\_part\_real\_part). imag\_part\_func(real\_part\_imag\_part). set\_name("real\_part","real\_part"))
- static `ex imag_part_evalf` (const `ex` &arg)
- static `ex imag_part_eval` (const `ex` &arg)
- static void `imag_part_print_latex` (const `ex` &arg, const `print_context` &c)
- static `ex imag_part_conjugate` (const `ex` &arg)
- static `ex imag_part_real_part` (const `ex` &arg)
- static `ex imag_part_imag_part` (const `ex` &arg)
- static `ex imag_part_expl_derivative` (const `ex` &arg, const `symbol` &s)

- [REGISTER\\_FUNCTION](#) ([imag\\_part\\_function](#), [eval\\_func\(imag\\_part\\_eval\)](#). [evalf\\_func\(imag\\_part\\_evalf\)](#). [expl\\_derivative\\_func\(imag\\_part\\_expl\\_derivative\)](#). [print\\_func< print\\_latex >\(imag\\_part\\_print\\_latex\)](#). [conjugate\\_func\(imag\\_part\\_conjugate\)](#). [real\\_part\\_func\(imag\\_part\\_real\\_part\)](#). [imag\\_part\\_func\(imag\\_part\\_imag\\_part\)](#). [set\\_name\("imag\\_part","imag\\_part"\)](#))
- static [ex abs\\_evalf](#) (const [ex](#) &arg)
- static [ex abs\\_eval](#) (const [ex](#) &arg)
- static [ex abs\\_expand](#) (const [ex](#) &arg, unsigned options)
- static [ex abs\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- static void [abs\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static void [abs\\_print\\_csrc\\_float](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex abs\\_conjugate](#) (const [ex](#) &arg)
- static [ex abs\\_real\\_part](#) (const [ex](#) &arg)
- static [ex abs\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex abs\\_power](#) (const [ex](#) &arg, const [ex](#) &exp)
- bool [abs\\_info](#) (const [ex](#) &arg, unsigned inf)
- [REGISTER\\_FUNCTION](#) ([abs](#), [eval\\_func\(abs\\_eval\)](#). [evalf\\_func\(abs\\_evalf\)](#). [expand\\_func\(abs\\_expand\)](#). [expl\\_derivative\\_func\(abs\\_expl\\_derivative\)](#). [info\\_func\(abs\\_info\)](#). [print\\_func< print\\_latex >\(abs\\_print\\_latex\)](#). [print\\_func< print\\_csrc\\_float >\(abs\\_print\\_csrc\\_float\)](#). [print\\_func< print\\_csrc\\_double >\(abs\\_print\\_csrc\\_double\)](#). [conjugate\\_func\(abs\\_conjugate\)](#). [real\\_part\\_func\(abs\\_real\\_part\)](#). [imag\\_part\\_func\(abs\\_imag\\_part\)](#). [power\\_func\(abs\\_power\)](#))
- static [ex step\\_evalf](#) (const [ex](#) &arg)
- static [ex step\\_eval](#) (const [ex](#) &arg)
- static [ex step\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)
- static [ex step\\_conjugate](#) (const [ex](#) &arg)
- static [ex step\\_real\\_part](#) (const [ex](#) &arg)
- static [ex step\\_imag\\_part](#) (const [ex](#) &arg)
- [REGISTER\\_FUNCTION](#) ([step](#), [eval\\_func\(step\\_eval\)](#). [evalf\\_func\(step\\_evalf\)](#). [series\\_func\(step\\_series\)](#). [conjugate\\_func\(step\\_conjugate\)](#). [real\\_part\\_func\(step\\_real\\_part\)](#). [imag\\_part\\_func\(step\\_imag\\_part\)](#))
- static [ex csgn\\_evalf](#) (const [ex](#) &arg)
- static [ex csgn\\_eval](#) (const [ex](#) &arg)
- static [ex csgn\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)
- static [ex csgn\\_conjugate](#) (const [ex](#) &arg)
- static [ex csgn\\_real\\_part](#) (const [ex](#) &arg)
- static [ex csgn\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex csgn\\_power](#) (const [ex](#) &arg, const [ex](#) &exp)
- [REGISTER\\_FUNCTION](#) ([csgn](#), [eval\\_func\(csgn\\_eval\)](#). [evalf\\_func\(csgn\\_evalf\)](#). [series\\_func\(csgn\\_series\)](#). [conjugate\\_func\(csgn\\_conjugate\)](#). [real\\_part\\_func\(csgn\\_real\\_part\)](#). [imag\\_part\\_func\(csgn\\_imag\\_part\)](#). [power\\_func\(csgn\\_power\)](#))
- static [ex eta\\_evalf](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex eta\\_eval](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex eta\\_series](#) (const [ex](#) &x, const [ex](#) &y, const [relational](#) &rel, int order, unsigned options)
- static [ex eta\\_conjugate](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex eta\\_real\\_part](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex eta\\_imag\\_part](#) (const [ex](#) &x, const [ex](#) &y)
- [REGISTER\\_FUNCTION](#) ([eta](#), [eval\\_func\(eta\\_eval\)](#). [evalf\\_func\(eta\\_evalf\)](#). [series\\_func\(eta\\_series\)](#). [latex\\_name\("\\eta"\)](#). [set\\_symmetry\(sy\\_symm\(0, 1\)\)](#). [conjugate\\_func\(eta\\_conjugate\)](#). [real\\_part\\_func\(eta\\_real\\_part\)](#). [imag\\_part\\_func\(eta\\_imag\\_part\)](#))
- static [ex Li2\\_evalf](#) (const [ex](#) &x)
- static [ex Li2\\_eval](#) (const [ex](#) &x)
- static [ex Li2\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex Li2\\_series](#) (const [ex](#) &x, const [relational](#) &rel, int order, unsigned options)
- static [ex Li2\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([Li2](#), [eval\\_func\(Li2\\_eval\)](#). [evalf\\_func\(Li2\\_evalf\)](#). [derivative\\_func\(Li2\\_deriv\)](#). [series\\_func\(Li2\\_series\)](#). [conjugate\\_func\(Li2\\_conjugate\)](#). [latex\\_name\("\\mathrm{Li}\\_2"\)](#))
- static [ex Li3\\_eval](#) (const [ex](#) &x)

- `REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`). `latex_name("\\mathrm{Li}_3")`)
  - static `ex zetaderiv_eval` (const `ex` &n, const `ex` &x)
  - static `ex zetaderiv_deriv` (const `ex` &n, const `ex` &x, unsigned `deriv_param`)
  - `REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`. `derivative_func(zetaderiv_deriv)`. `latex_name("\\zeta^\\prime")`)
  - static `ex factorial_evalf` (const `ex` &x)
  - static `ex factorial_eval` (const `ex` &x)
  - static void `factorial_print_dflt_latex` (const `ex` &x, const `print_context` &c)
  - static `ex factorial_conjugate` (const `ex` &x)
  - static `ex factorial_real_part` (const `ex` &x)
  - static `ex factorial_imag_part` (const `ex` &x)
  - `REGISTER_FUNCTION` (`factorial`, `eval_func(factorial_eval)`. `evalf_func(factorial_evalf)`. `print_func<print_dflt>(factorial_print_dflt_latex)`. `print_func<print_latex>(factorial_print_dflt_latex)`. `conjugate_func(factorial_conjugate)`. `real_part_func(factorial_real_part)`. `imag_part_func(factorial_imag_part)`)
  - static `ex binomial_evalf` (const `ex` &x, const `ex` &y)
  - static `ex binomial_sym` (const `ex` &x, const `numeric` &y)
  - static `ex binomial_eval` (const `ex` &x, const `ex` &y)
  - static `ex binomial_conjugate` (const `ex` &x, const `ex` &y)
  - static `ex binomial_real_part` (const `ex` &x, const `ex` &y)
  - static `ex binomial_imag_part` (const `ex` &x, const `ex` &y)
  - `REGISTER_FUNCTION` (`binomial`, `eval_func(binomial_eval)`. `evalf_func(binomial_evalf)`. `conjugate_func(binomial_conjugate)`. `real_part_func(binomial_real_part)`. `imag_part_func(binomial_imag_part)`)
  - static `ex Order_eval` (const `ex` &x)
  - static `ex Order_series` (const `ex` &x, const `relational` &r, int `order`, unsigned `options`)
  - static `ex Order_conjugate` (const `ex` &x)
  - static `ex Order_real_part` (const `ex` &x)
  - static `ex Order_imag_part` (const `ex` &x)
  - static `ex Order_power` (const `ex` &x, const `ex` &e)
  - static `ex Order_expl_derivative` (const `ex` &arg, const `symbol` &s)
  - `REGISTER_FUNCTION` (`Order`, `eval_func(Order_eval)`. `series_func(Order_series)`. `latex_name("\\mathcal{O}")`. `expl_derivative_func(Order_expl_derivative)`. `conjugate_func(Order_conjugate)`. `real_part_func(Order_real_part)`. `imag_part_func(Order_imag_part)`. `power_func(Order_power)`)
  - `ex Isolve` (const `ex` &eqns, const `ex` &symbols, unsigned `options=solve_algo::automatic`)
- Factorial function.*
- const `numeric fsolve` (const `ex` &f, const `symbol` &x, const `numeric` &x1, const `numeric` &x2)
- Find a real root of real-valued function f(x) numerically within a given interval.*
- template<typename T1>  
  `function zeta` (const T1 &p1)
  - template<typename T1, typename T2>  
  `function zeta` (const T1 &p1, const T2 &p2)
  - template<> bool `is_the_function<zeta_SERIAL>` (const `ex` &x)
  - template<typename T1, typename T2>  
  `function G` (const T1 &x, const T2 &y)
  - template<typename T1, typename T2, typename T3>  
  `function G` (const T1 &x, const T2 &s, const T3 &y)
  - template<> bool `is_the_function<G_SERIAL>` (const `ex` &x)
  - template<typename T1>  
  `function psi` (const T1 &p1)
  - template<typename T1, typename T2>  
  `function psi` (const T1 &p1, const T2 &p2)
  - template<> bool `is_the_function<psi_SERIAL>` (const `ex` &x)
  - template<typename T1, typename T2>  
  `function iterated_integral` (const T1 &kernel\_lst, const T2 &lambda)
  - template<typename T1, typename T2, typename T3>  
  `function iterated_integral` (const T1 &kernel\_lst, const T2 &lambda, const T3 &N\_trunc)



- `template<> bool is_the_function< iterated_integral_SERIAL > (const ex &x)`
- `bool is_order_function (const ex &e)`  
*Check whether a function is the Order  $O(n)$  function.*
- `ex convert_H_to_Li (const ex &parameterlst, const ex &arg)`  
*Converts a given list containing parameters for  $H$  in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*
- `static ex EllipticK_evalf (const ex &k)`
- `static ex EllipticK_eval (const ex &k)`
- `static ex EllipticK_deriv (const ex &k, unsigned deriv_param)`
- `static ex EllipticK_series (const ex &k, const relational &rel, int order, unsigned options)`
- `static void EllipticK_print_latex (const ex &k, const print_context &c)`
- `REGISTER_FUNCTION (EllipticK, evalf_func(EllipticK_evalf). eval_func(EllipticK_eval). derivative_↵  
func(EllipticK_deriv). series_func(EllipticK_series). print_func< print_latex >(EllipticK_print_latex). do_↵  
not_evalf_params())`
- `static ex EllipticE_evalf (const ex &k)`
- `static ex EllipticE_eval (const ex &k)`
- `static ex EllipticE_deriv (const ex &k, unsigned deriv_param)`
- `static ex EllipticE_series (const ex &k, const relational &rel, int order, unsigned options)`
- `static void EllipticE_print_latex (const ex &k, const print_context &c)`
- `REGISTER_FUNCTION (EllipticE, evalf_func(EllipticE_evalf). eval_func(EllipticE_eval). derivative_↵  
func(EllipticE_deriv). series_func(EllipticE_series). print_func< print_latex >(EllipticE_print_latex). do_↵  
not_evalf_params())`
- `static ex iterated_integral_evalf_impl (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex iterated_integral2_evalf (const ex &kernel_lst, const ex &lambda)`
- `static ex iterated_integral3_evalf (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex iterated_integral2_eval (const ex &kernel_lst, const ex &lambda)`
- `static ex iterated_integral3_eval (const ex &kernel_lst, const ex &lambda, const ex &N_trunc)`
- `static ex lgamma_evalf (const ex &x)`
- `static ex lgamma_eval (const ex &x)`  
*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- `static ex lgamma_deriv (const ex &x, unsigned deriv_param)`
- `static ex lgamma_series (const ex &arg, const relational &rel, int order, unsigned options)`
- `static ex lgamma_conjugate (const ex &x)`
- `REGISTER_FUNCTION (lgamma, eval_func(lgamma_eval). evalf_func(lgamma_evalf). derivative_↵  
func(lgamma_deriv). series_func(lgamma_series). conjugate_func(lgamma_conjugate). latex_name("\\log  
\\Gamma"))`
- `static ex tgamma_evalf (const ex &x)`
- `static ex tgamma_eval (const ex &x)`  
*Evaluation of  $tgamma(x)$ , the true Gamma function.*
- `static ex tgamma_deriv (const ex &x, unsigned deriv_param)`
- `static ex tgamma_series (const ex &arg, const relational &rel, int order, unsigned options)`
- `static ex tgamma_conjugate (const ex &x)`
- `REGISTER_FUNCTION (tgamma, eval_func(tgamma_eval). evalf_func(tgamma_evalf). derivative_↵  
func(tgamma_deriv). series_func(tgamma_series). conjugate_func(tgamma_conjugate). latex_↵  
name("\\Gamma"))`
- `static ex beta_evalf (const ex &x, const ex &y)`
- `static ex beta_eval (const ex &x, const ex &y)`
- `static ex beta_deriv (const ex &x, const ex &y, unsigned deriv_param)`
- `static ex beta_series (const ex &arg1, const ex &arg2, const relational &rel, int order, unsigned options)`
- `REGISTER_FUNCTION (beta, eval_func(beta_eval). evalf_func(beta_evalf). derivative_func(beta_deriv).  
series_func(beta_series). latex_name("\\mathrm{B}"). set_symmetry(sy_symm(0, 1)))`
- `static ex psi1_evalf (const ex &x)`
- `static ex psi1_eval (const ex &x)`  
*Evaluation of digamma-function  $\psi(x)$ .*

- static `ex psi1_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex psi1_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex psi2_evalf` (const `ex` &n, const `ex` &x)
- static `ex psi2_eval` (const `ex` &n, const `ex` &x)

*Evaluation of polygamma-function  $\psi(n,x)$ .*

- static `ex psi2_deriv` (const `ex` &n, const `ex` &x, unsigned deriv\_param)
- static `ex psi2_series` (const `ex` &n, const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex G2_evalf` (const `ex` &x\_, const `ex` &y)
- static `ex G2_eval` (const `ex` &x\_, const `ex` &y)
- static `ex G3_evalf` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex G3_eval` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex Li_evalf` (const `ex` &m\_, const `ex` &x\_)
- static `ex Li_eval` (const `ex` &m\_, const `ex` &x\_)
- static `ex Li_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex Li_deriv` (const `ex` &m\_, const `ex` &x\_, unsigned deriv\_param)
- static void `Li_print_latex` (const `ex` &m\_, const `ex` &x\_, const `print_context` &c)
- `REGISTER_FUNCTION` (Li, evalf\_func(`Li_evalf`). eval\_func(`Li_eval`). series\_func(`Li_series`). derivative\_↔  
func(`Li_deriv`). print\_func< `print_latex` >(`Li_print_latex`). do\_not\_evalf\_params())
- static `ex S_evalf` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex S_eval` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex S_series` (const `ex` &n, const `ex` &p, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex S_deriv` (const `ex` &n, const `ex` &p, const `ex` &x, unsigned deriv\_param)
- static void `S_print_latex` (const `ex` &n, const `ex` &p, const `ex` &x, const `print_context` &c)
- `REGISTER_FUNCTION` (S, evalf\_func(`S_evalf`). eval\_func(`S_eval`). series\_func(`S_series`). derivative\_↔  
func(`S_deriv`). print\_func< `print_latex` >(`S_print_latex`). do\_not\_evalf\_params())
- static `ex H_evalf` (const `ex` &x1, const `ex` &x2)
- static `ex H_eval` (const `ex` &m\_, const `ex` &x)
- static `ex H_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex H_deriv` (const `ex` &m\_, const `ex` &x, unsigned deriv\_param)
- static void `H_print_latex` (const `ex` &m\_, const `ex` &x, const `print_context` &c)
- `REGISTER_FUNCTION` (H, evalf\_func(`H_evalf`). eval\_func(`H_eval`). series\_func(`H_series`). derivative\_↔  
func(`H_deriv`). print\_func< `print_latex` >(`H_print_latex`). do\_not\_evalf\_params())
- static `ex zeta1_evalf` (const `ex` &x)
- static `ex zeta1_eval` (const `ex` &m)
- static `ex zeta1_deriv` (const `ex` &m, unsigned deriv\_param)
- static void `zeta1_print_latex` (const `ex` &m\_, const `print_context` &c)
- static `ex zeta2_evalf` (const `ex` &x, const `ex` &s)
- static `ex zeta2_eval` (const `ex` &m, const `ex` &s\_)
- static `ex zeta2_deriv` (const `ex` &m, const `ex` &s, unsigned deriv\_param)
- static void `zeta2_print_latex` (const `ex` &m\_, const `ex` &s\_, const `print_context` &c)
- static `ex exp_evalf` (const `ex` &x)
- static `ex exp_eval` (const `ex` &x)
- static `ex exp_expand` (const `ex` &arg, unsigned options)
- static `ex exp_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex exp_real_part` (const `ex` &x)
- static `ex exp_imag_part` (const `ex` &x)
- static `ex exp_conjugate` (const `ex` &x)
- static `ex exp_power` (const `ex` &x, const `ex` &a)
- static bool `exp_info` (const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (exp, eval\_func(`exp_eval`). evalf\_func(`exp_evalf`). info\_func(`exp_info`). expand\_↔  
func(`exp_expand`). derivative\_func(`exp_deriv`). real\_part\_func(`exp_real_part`). imag\_part\_func(`exp_imag_part`).  
conjugate\_func(`exp_conjugate`). power\_func(`exp_power`). latex\_name("\\exp"))
- static `ex log_evalf` (const `ex` &x)
- static `ex log_eval` (const `ex` &x)
- static `ex log_deriv` (const `ex` &x, unsigned deriv\_param)



- static `ex log_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex log_real_part` (const `ex` &x)
- static `ex log_imag_part` (const `ex` &x)
- static `ex log_expand` (const `ex` &arg, unsigned options)
- static `ex log_conjugate` (const `ex` &x)
- static bool `log_info` (const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (`log`, `eval_func(log_eval)`, `evalf_func(log_evalf)`, `info_func(log_info)`, `expand_func(log_expand)`, `derivative_func(log_deriv)`, `series_func(log_series)`, `real_part_func(log_real_part)`, `imag_part_func(log_imag_part)`, `conjugate_func(log_conjugate)`, `latex_name("\\ln")`)
- static `ex sin_evalf` (const `ex` &x)
- static `ex sin_eval` (const `ex` &x)
- static `ex sin_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex sin_real_part` (const `ex` &x)
- static `ex sin_imag_part` (const `ex` &x)
- static `ex sin_conjugate` (const `ex` &x)
- static bool `trig_info` (const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (`sin`, `eval_func(sin_eval)`, `evalf_func(sin_evalf)`, `info_func(trig_info)`, `derivative_func(sin_deriv)`, `real_part_func(sin_real_part)`, `imag_part_func(sin_imag_part)`, `conjugate_func(sin_conjugate)`, `latex_name("\\sin")`)
- static `ex cos_evalf` (const `ex` &x)
- static `ex cos_eval` (const `ex` &x)
- static `ex cos_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex cos_real_part` (const `ex` &x)
- static `ex cos_imag_part` (const `ex` &x)
- static `ex cos_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`cos`, `eval_func(cos_eval)`, `info_func(trig_info)`, `evalf_func(cos_evalf)`, `derivative_func(cos_deriv)`, `real_part_func(cos_real_part)`, `imag_part_func(cos_imag_part)`, `conjugate_func(cos_conjugate)`, `latex_name("\\cos")`)
- static `ex tan_evalf` (const `ex` &x)
- static `ex tan_eval` (const `ex` &x)
- static `ex tan_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex tan_real_part` (const `ex` &x)
- static `ex tan_imag_part` (const `ex` &x)
- static `ex tan_series` (const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex tan_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`tan`, `eval_func(tan_eval)`, `evalf_func(tan_evalf)`, `info_func(trig_info)`, `derivative_func(tan_deriv)`, `series_func(tan_series)`, `real_part_func(tan_real_part)`, `imag_part_func(tan_imag_part)`, `conjugate_func(tan_conjugate)`, `latex_name("\\tan")`)
- static `ex asin_evalf` (const `ex` &x)
- static `ex asin_eval` (const `ex` &x)
- static `ex asin_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex asin_conjugate` (const `ex` &x)
- static bool `asin_info` (const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (`asin`, `eval_func(asin_eval)`, `evalf_func(asin_evalf)`, `info_func(asin_info)`, `derivative_func(asin_deriv)`, `conjugate_func(asin_conjugate)`, `latex_name("\\arcsin")`)
- static `ex acos_evalf` (const `ex` &x)
- static `ex acos_eval` (const `ex` &x)
- static `ex acos_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex acos_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`acos`, `eval_func(acos_eval)`, `evalf_func(acos_evalf)`, `info_func(asin_info)`, `derivative_func(acos_deriv)`, `conjugate_func(acos_conjugate)`, `latex_name("\\arccos")`)
- static `ex atan_evalf` (const `ex` &x)
- static `ex atan_eval` (const `ex` &x)
- static `ex atan_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex atan_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)

- static [ex atan\\_conjugate](#) (const [ex](#) &x)
- static bool [atan\\_info](#) (const [ex](#) &x, unsigned inf)
- [REGISTER\\_FUNCTION](#) ([atan](#), [eval\\_func\(atan\\_eval\)](#). [evalf\\_func\(atan\\_evalf\)](#). [info\\_func\(atan\\_info\)](#). [derivative\\_func\(atan\\_deriv\)](#). [series\\_func\(atan\\_series\)](#). [conjugate\\_func\(atan\\_conjugate\)](#). [latex\\_name\("\\arctan"\)](#))
- static [ex atan2\\_evalf](#) (const [ex](#) &y, const [ex](#) &x)
- static [ex atan2\\_eval](#) (const [ex](#) &y, const [ex](#) &x)
- static [ex atan2\\_deriv](#) (const [ex](#) &y, const [ex](#) &x, unsigned deriv\_param)
- static bool [atan2\\_info](#) (const [ex](#) &y, const [ex](#) &x, unsigned inf)
- [REGISTER\\_FUNCTION](#) ([atan2](#), [eval\\_func\(atan2\\_eval\)](#). [evalf\\_func\(atan2\\_evalf\)](#). [info\\_func\(atan2\\_info\)](#). [evalf\\_func\(atan2\\_evalf\)](#). [derivative\\_func\(atan2\\_deriv\)](#))
- static [ex sinh\\_evalf](#) (const [ex](#) &x)
- static [ex sinh\\_eval](#) (const [ex](#) &x)
- static [ex sinh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex sinh\\_real\\_part](#) (const [ex](#) &x)
- static [ex sinh\\_imag\\_part](#) (const [ex](#) &x)
- static [ex sinh\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([sinh](#), [eval\\_func\(sinh\\_eval\)](#). [evalf\\_func\(sinh\\_evalf\)](#). [info\\_func\(atan\\_info\)](#). [derivative\\_func\(sinh\\_deriv\)](#). [real\\_part\\_func\(sinh\\_real\\_part\)](#). [imag\\_part\\_func\(sinh\\_imag\\_part\)](#). [conjugate\\_func\(sinh\\_conjugate\)](#). [latex\\_name\("\\sinh"\)](#))
- static [ex cosh\\_evalf](#) (const [ex](#) &x)
- static [ex cosh\\_eval](#) (const [ex](#) &x)
- static [ex cosh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex cosh\\_real\\_part](#) (const [ex](#) &x)
- static [ex cosh\\_imag\\_part](#) (const [ex](#) &x)
- static [ex cosh\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([cosh](#), [eval\\_func\(cosh\\_eval\)](#). [evalf\\_func\(cosh\\_evalf\)](#). [info\\_func\(exp\\_info\)](#). [derivative\\_func\(cosh\\_deriv\)](#). [real\\_part\\_func\(cosh\\_real\\_part\)](#). [imag\\_part\\_func\(cosh\\_imag\\_part\)](#). [conjugate\\_func\(cosh\\_conjugate\)](#). [latex\\_name\("\\cosh"\)](#))
- static [ex tanh\\_evalf](#) (const [ex](#) &x)
- static [ex tanh\\_eval](#) (const [ex](#) &x)
- static [ex tanh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex tanh\\_series](#) (const [ex](#) &x, const [relational](#) &rel, int order, unsigned options)
- static [ex tanh\\_real\\_part](#) (const [ex](#) &x)
- static [ex tanh\\_imag\\_part](#) (const [ex](#) &x)
- static [ex tanh\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([tanh](#), [eval\\_func\(tanh\\_eval\)](#). [evalf\\_func\(tanh\\_evalf\)](#). [info\\_func\(atan\\_info\)](#). [derivative\\_func\(tanh\\_deriv\)](#). [series\\_func\(tanh\\_series\)](#). [real\\_part\\_func\(tanh\\_real\\_part\)](#). [imag\\_part\\_func\(tanh\\_imag\\_part\)](#). [conjugate\\_func\(tanh\\_conjugate\)](#). [latex\\_name\("\\tanh"\)](#))
- static [ex asinh\\_evalf](#) (const [ex](#) &x)
- static [ex asinh\\_eval](#) (const [ex](#) &x)
- static [ex asinh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex asinh\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([asinh](#), [eval\\_func\(asinh\\_eval\)](#). [evalf\\_func\(asinh\\_evalf\)](#). [info\\_func\(atan\\_info\)](#). [derivative\\_func\(asinh\\_deriv\)](#). [conjugate\\_func\(asinh\\_conjugate\)](#))
- static [ex acosh\\_evalf](#) (const [ex](#) &x)
- static [ex acosh\\_eval](#) (const [ex](#) &x)
- static [ex acosh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex acosh\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) ([acosh](#), [eval\\_func\(acosh\\_eval\)](#). [evalf\\_func\(acosh\\_evalf\)](#). [info\\_func\(asin\\_info\)](#). [derivative\\_func\(acosh\\_deriv\)](#). [conjugate\\_func\(acosh\\_conjugate\)](#))
- static [ex atanh\\_evalf](#) (const [ex](#) &x)
- static [ex atanh\\_eval](#) (const [ex](#) &x)
- static [ex atanh\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex atanh\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)

- static `ex atanh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`atanh`, `eval_func(atanh_eval)`, `evalf_func(atanh_evalf)`, `info_func(asin_info)`, `derivative_func(atanh_deriv)`, `series_func(atanh_series)`, `conjugate_func(atanh_conjugate)`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integral`, `basic`, `print_func< print_dflt >(&integral::do_print)`, `print_func< print_python >(&integral::do_print)`, `print_func< print_latex >(&integral::do_print_latex)`) `integral`
- `ex subsvalue` (const `ex` &var, const `ex` &value, const `ex` &fun)
- `ex adaptivesimpson` (const `ex` &x, const `ex` &a\_in, const `ex` &b\_in, const `ex` &f, const `ex` &error)
 

*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- `GINAC_BIND_UNARCHIVER` (`integral`)
- `GINAC_DECLARE_UNARCHIVER` (`integral`)
- `ex ifactor` (const `numeric` &n)
 

*Returns the decomposition of the positive integer n into prime numbers in the form  $l_1^{p_1} \dots l_r^{p_r}$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .*
- `bool is_discriminant_of_quadratic_number_field` (const `numeric` &n)
 

*Returns true if the integer n is either one or the discriminant of a quadratic number field.*
- `numeric kronecker_symbol` (const `numeric` &a, const `numeric` &n)
 

*Returns the Kronecker symbol a: integer n: integer.*
- `numeric primitive_dirichlet_character` (const `numeric` &n, const `numeric` &a)
 

*Defines a primitive Dirichlet character through the Kronecker symbol.*
- `numeric dirichlet_character` (const `numeric` &n, const `numeric` &a, const `numeric` &N)
 

*Defines a Dirichlet character through the Kronecker symbol.*
- `numeric generalised_Bernoulli_number` (const `numeric` &k, const `numeric` &b)
 

*The generalised Bernoulli number.*
- `ex Bernoulli_polynomial` (const `numeric` &k, const `ex` &x)
 

*The Bernoulli polynomials.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integration_kernel`, `basic`, `print_func< print_context >(&integration_kernel::do_print)`) `integration_kernel`
- `GINAC_BIND_UNARCHIVER` (`integration_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`basic_log_kernel`, `integration_kernel`, `print_func< print_context >(&basic_log_kernel::do_print)`) `basic_log_kernel`
- `GINAC_BIND_UNARCHIVER` (`basic_log_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`multiple_polylog_kernel`, `integration_kernel`, `print_func< print_context >(&multiple_polylog_kernel::do_print)`) `multiple_polylog_kernel`
- `GINAC_BIND_UNARCHIVER` (`multiple_polylog_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`ELi_kernel`, `integration_kernel`, `print_func< print_context >(&ELi_kernel::do_print)`) `ELi_kernel`
- `GINAC_BIND_UNARCHIVER` (`ELi_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Ebar_kernel`, `integration_kernel`, `print_func< print_context >(&Ebar_kernel::do_print)`) `Ebar_kernel`
- `GINAC_BIND_UNARCHIVER` (`Ebar_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Kronecker_dtau_kernel`, `integration_kernel`, `print_func< print_context >(&Kronecker_dtau_kernel::do_print)`) `Kronecker_dtau_kernel`
- `GINAC_BIND_UNARCHIVER` (`Kronecker_dtau_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Kronecker_dz_kernel`, `integration_kernel`, `print_func< print_context >(&Kronecker_dz_kernel::do_print)`) `Kronecker_dz_kernel`
- `GINAC_BIND_UNARCHIVER` (`Kronecker_dz_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Eisenstein_kernel`, `integration_kernel`, `print_func< print_context >(&Eisenstein_kernel::do_print)`) `Eisenstein_kernel`
- `GINAC_BIND_UNARCHIVER` (`Eisenstein_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`Eisenstein_h_kernel`, `integration_kernel`, `print_func< print_context >(&Eisenstein_h_kernel::do_print)`) `Eisenstein_h_kernel`
- `GINAC_BIND_UNARCHIVER` (`Eisenstein_h_kernel`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`modular_form_kernel`, `integration_kernel`, `print_func< print_context >(&modular_form_kernel::do_print)`) `modular_form_kernel`

- [GINAC\\_BIND\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([user\\_defined\\_kernel](#), [integration\\_kernel](#), [print\\_func](#)<[print\\_context](#) >(&[user\\_defined\\_kernel](#)::do\_print)) [user\\_defined\\_kernel](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([integration\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([basic\\_log\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([multiple\\_polylog\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([ELi\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Ebar\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([lst](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([matrix](#), [basic](#), [print\\_func](#)<[print\\_context](#) >(&[matrix](#)::do\_print). [print\\_func](#)<[print\\_latex](#) >(&[matrix](#)::do\_print\_latex). [print\\_func](#)<[print\\_tree](#) >(&[matrix](#)::do\_print\_tree). [print\\_func](#)<[print\\_python\\_repr](#) >(&[matrix](#)::do\_print\_python\_repr)) [matrix](#)

*Default ctor.*

- [GINAC\\_BIND\\_UNARCHIVER](#) ([matrix](#))
- [ex lst\\_to\\_matrix](#) (const [lst](#) &l)

*Convert list of lists to matrix.*

- [ex diag\\_matrix](#) (const [lst](#) &l)

*Convert list of diagonal elements to matrix.*

- [ex diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex unit\\_matrix](#) (unsigned r, unsigned c)

*Create an r times c unit matrix.*

- [ex symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_base\_name)

*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*

- [ex reduced\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned c)

*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*

- [ex sub\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned nr, unsigned c, unsigned nc)

*Return the nr times nc submatrix starting at position r, c of matrix m.*

- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t nops](#) (const [matrix](#) &m)
- [ex expand](#) (const [matrix](#) &m, unsigned options=0)
- [ex evalf](#) (const [matrix](#) &m)
- unsigned [rows](#) (const [matrix](#) &m)
- unsigned [cols](#) (const [matrix](#) &m)
- [matrix transpose](#) (const [matrix](#) &m)
- [ex determinant](#) (const [matrix](#) &m, unsigned options=[determinant\\_algo::automatic](#))
- [ex trace](#) (const [matrix](#) &m)
- [ex charpoly](#) (const [matrix](#) &m, const [ex](#) &lambda)
- [matrix inverse](#) (const [matrix](#) &m)
- [matrix inverse](#) (const [matrix](#) &m, unsigned algo)
- unsigned [rank](#) (const [matrix](#) &m)
- unsigned [rank](#) (const [matrix](#) &m, unsigned [solve\\_algo](#))
- [ex unit\\_matrix](#) (unsigned x)

*Create a x times x unit matrix.*

- [ex symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name)

Create an  $r$  times  $c$  matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`mul`, `expairseq`, `print_func`< `print_context` >(&`mul::do_print`). `print_func`< `print_latex` >(&`mul::do_print_latex`). `print_func`< `print_csrc` >(&`mul::do_print_csrc`). `print_func`< `print_tree` >(&`mul::do_print_tree`). `print_func`< `print_python_repr` >(&`mul::do_print_python_repr`) `mul`
- `bool tryfactsubs` (const `ex` &`origfactor`, const `ex` &`patternfactor`, int &`nummatches`, `exmap` &`repls`)
- `bool algebraic_match_mul_with_mul` (const `mul` &`e`, const `ex` &`pat`, `exmap` &`repls`, int `factor`, int &`nummatches`, const `std::vector`< `bool` > &`subsed`, `std::vector`< `bool` > &`matched`)

Checks whether  $e$  matches to the pattern  $pat$  and the (possibly to be updated) list of replacements  $repls$ .

- `GINAC_BIND_UNARCHIVER` (`mul`)
- `GINAC_DECLARE_UNARCHIVER` (`mul`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`ncmul`, `exprseq`, `print_func`< `print_context` >(&`ncmul::do_print`). `print_func`< `print_tree` >(&`ncmul::do_print_tree`). `print_func`< `print_csrc` >(&`ncmul::do_print_csrc`). `print_func`< `print_python_repr` >(&`ncmul::do_print_csrc`)) `ncmul`
- `ex reeval_ncmul` (const `exvector` &`v`)
- `ex hold_ncmul` (const `exvector` &`v`)
- `GINAC_BIND_UNARCHIVER` (`ncmul`)
- `GINAC_DECLARE_UNARCHIVER` (`ncmul`)
- static `bool get_first_symbol` (const `ex` &`e`, `ex` &`x`)

Return pointer to first symbol found in expression.

- static void `add_symbol` (const `ex` &`s`, `sym_desc_vec` &`v`)
- static void `collect_symbols` (const `ex` &`e`, `sym_desc_vec` &`v`)
- static void `get_symbol_stats` (const `ex` &`a`, const `ex` &`b`, `sym_desc_vec` &`v`)

Collect statistical information about symbols in polynomials.

- static `numeric lcmcoeff` (const `ex` &`e`, const `numeric` &`l`)
- static `numeric lcm_of_coefficients_denominators` (const `ex` &`e`)

Compute LCM of denominators of coefficients of a polynomial.

- static `ex multiply_lcm` (const `ex` &`e`, const `numeric` &`lcm`)

Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.

- `ex quo` (const `ex` &`a`, const `ex` &`b`, const `ex` &`x`, bool `check_args`)

Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .

- `ex rem` (const `ex` &`a`, const `ex` &`b`, const `ex` &`x`, bool `check_args`)

Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .

- `ex decomp_rational` (const `ex` &`a`, const `ex` &`x`)

Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .

- `ex prem` (const `ex` &`a`, const `ex` &`b`, const `ex` &`x`, bool `check_args`)

Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .

- `ex sprem` (const `ex` &`a`, const `ex` &`b`, const `ex` &`x`, bool `check_args`)

Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .

- `bool divide` (const `ex` &`a`, const `ex` &`b`, `ex` &`q`, bool `check_args`)

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .

- static `bool divide_in_z` (const `ex` &`a`, const `ex` &`b`, `ex` &`q`, `sym_desc_vec::const_iterator` `var`)

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .

- static `ex sr_gcd` (const `ex` &`a`, const `ex` &`b`, `sym_desc_vec::const_iterator` `var`)

Compute GCD of multivariate polynomials using the subresultant PRS algorithm.

- static `ex interpolate` (const `ex` &`gamma`, const `numeric` &`xi`, const `ex` &`x`, int `degree_hint`=1)

$\xi$ -adic polynomial interpolation

- static `bool heur_gcd_z` (`ex` &`res`, const `ex` &`a`, const `ex` &`b`, `ex` \*`ca`, `ex` \*`cb`, `sym_desc_vec::const_iterator` `var`)

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

- static `bool heur_gcd` (`ex` &`res`, const `ex` &`a`, const `ex` &`b`, `ex` \*`ca`, `ex` \*`cb`, `sym_desc_vec::const_iterator` `var`)

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

- static `ex gcd_pf_pow` (const `ex` &`a`, const `ex` &`b`, `ex` \*`ca`, `ex` \*`cb`)

- static `ex gcd_pf_mul` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex gcd` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, bool check\_args, unsigned options)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- static `ex gcd_pf_pow_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex lcm` (const `ex` &a, const `ex` &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- static `epvector sqrfree_yun` (const `ex` &a, const `symbol` &x)  
*Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
- `ex sqrfree` (const `ex` &a, const `lst` &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- `ex sqrfree_parfrac` (const `ex` &a, const `symbol` &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- static `ex replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex replace_with_symbol` (const `ex` &e, `exmap` &repl)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex frac_cancel` (const `ex` &n, const `ex` &d)  
*Fraction cancellation.*
- static `ex find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)  
*Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
- `ex collect_common_factors` (const `ex` &e)  
*Collect common factors in sums.*
- `ex resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)  
*Resultant of two expressions  $e1, e2$  with respect to symbol  $s$ .*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`numeric`, `basic`, print\_func< `print_context` >(&numeric::do\_print), print\_func< `print_latex` >(&numeric::do\_print\_latex), print\_func< `print_csrc` >(&numeric::do\_print\_csrc), print\_func< `print_csrc_cl_N` >(&numeric::do\_print\_csrc\_cl\_N), print\_func< `print_tree` >(&numeric::do\_print\_tree), print\_func< `print_python_repr` >(&numeric::do\_print\_python\_repr)) `numeric`  
*default ctor.*
- static const `cln::cl_F make_real_float` (const `cln::cl_idcoded_float` &dec)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const `cln::cl_F read_real_float` (std::istream &s)  
*Read serialized floating point number.*
- `GINAC_BIND_UNARCHIVER` (`numeric`)
- static void `write_real_float` (std::ostream &s, const `cln::cl_R` &n)
- static void `print_real_number` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void `print_integer_csrc` (const `print_context` &c, const `cln::cl_I` &x)  
*Helper function to print integer number in C++ source format.*
- static void `print_real_csrc` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print real number in C++ source format.*
- template<typename T1, typename T2>  
static bool `coerce` (T1 &dst, const T2 &arg)
- template<> bool `coerce`< `int`, `cln::cl_I` > (int &dst, const `cln::cl_I` &arg)  
*Check if CLN integer can be converted into int.*
- template<> bool `coerce`< `unsigned int`, `cln::cl_I` > (unsigned int &dst, const `cln::cl_I` &arg)
- static void `print_real_cl_N` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print real number in C++ source format using  $cl\_N$  types.*
- const `numeric exp` (const `numeric` &x)  
*Exponential function.*
- const `numeric log` (const `numeric` &x)



- *Natural logarithm.*
- const [numeric sin](#) (const [numeric](#) &x)
- *Numeric sine (trigonometric function).*
- const [numeric cos](#) (const [numeric](#) &x)
- *Numeric cosine (trigonometric function).*
- const [numeric tan](#) (const [numeric](#) &x)
- *Numeric tangent (trigonometric function).*
- const [numeric asin](#) (const [numeric](#) &x)
- *Numeric inverse sine (trigonometric function).*
- const [numeric acos](#) (const [numeric](#) &x)
- *Numeric inverse cosine (trigonometric function).*
- const [numeric atan](#) (const [numeric](#) &x)
- *Numeric arcustangent.*
- const [numeric atan](#) (const [numeric](#) &y, const [numeric](#) &x)
- *Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const [numeric sinh](#) (const [numeric](#) &x)
- *Numeric hyperbolic sine (trigonometric function).*
- const [numeric cosh](#) (const [numeric](#) &x)
- *Numeric hyperbolic cosine (trigonometric function).*
- const [numeric tanh](#) (const [numeric](#) &x)
- *Numeric hyperbolic tangent (trigonometric function).*
- const [numeric asinh](#) (const [numeric](#) &x)
- *Numeric inverse hyperbolic sine (trigonometric function).*
- const [numeric acosh](#) (const [numeric](#) &x)
- *Numeric inverse hyperbolic cosine (trigonometric function).*
- const [numeric atanh](#) (const [numeric](#) &x)
- *Numeric inverse hyperbolic tangent (trigonometric function).*
- static [cln::cl\\_N Li2\\_series](#) (const [cln::cl\\_N](#) &x, const [cln::float\\_format\\_t](#) &prec)
- *Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static [cln::cl\\_N Li2\\_projection](#) (const [cln::cl\\_N](#) &x, const [cln::float\\_format\\_t](#) &prec)
- *Folds Li2's argument inside a small rectangle to enhance convergence.*
- const [cln::cl\\_N Li2\\_](#) (const [cln::cl\\_N](#) &value)
- *Numeric evaluation of Dilogarithm.*
- const [numeric Li2](#) (const [numeric](#) &x)
- const [numeric zeta](#) (const [numeric](#) &x)
- *Numeric evaluation of Riemann's Zeta function.*
- static [cln::float\\_format\\_t guess\\_precision](#) (const [cln::cl\\_N](#) &x)
- const [cln::cl\\_N lgamma](#) (const [cln::cl\\_N](#) &x)
- *The Gamma function.*
- const [numeric lgamma](#) (const [numeric](#) &x)
- const [cln::cl\\_N tgamma](#) (const [cln::cl\\_N](#) &x)
- const [numeric tgamma](#) (const [numeric](#) &x)
- const [numeric psi](#) (const [numeric](#) &x)
- *The psi function (aka polygamma function).*
- const [numeric psi](#) (const [numeric](#) &n, const [numeric](#) &x)
- *The psi functions (aka polygamma functions).*
- const [numeric factorial](#) (const [numeric](#) &n)
- *Factorial combinatorial function.*
- const [numeric doublefactorial](#) (const [numeric](#) &n)
- *The double factorial combinatorial function.*
- const [numeric binomial](#) (const [numeric](#) &n, const [numeric](#) &k)

*The Binomial coefficients.*

- const [numeric bernoulli](#) (const [numeric](#) &nn)

*Bernoulli number.*

- const [numeric fibonacci](#) (const [numeric](#) &n)

*Fibonacci number.*

- const [numeric abs](#) (const [numeric](#) &x)

*Absolute value.*

- const [numeric mod](#) (const [numeric](#) &a, const [numeric](#) &b)

*Modulus (in positive representation).*

- const [numeric smod](#) (const [numeric](#) &a\_, const [numeric](#) &b\_)

*Modulus (in symmetric representation).*

- const [numeric irem](#) (const [numeric](#) &a, const [numeric](#) &b)

*Numeric integer remainder.*

- const [numeric irem](#) (const [numeric](#) &a, const [numeric](#) &b, [numeric](#) &q)

*Numeric integer remainder.*

- const [numeric iquo](#) (const [numeric](#) &a, const [numeric](#) &b)

*Numeric integer quotient.*

- const [numeric iquo](#) (const [numeric](#) &a, const [numeric](#) &b, [numeric](#) &r)

*Numeric integer quotient.*

- const [numeric gcd](#) (const [numeric](#) &a, const [numeric](#) &b)

*Greatest Common Divisor.*

- const [numeric lcm](#) (const [numeric](#) &a, const [numeric](#) &b)

*Least Common Multiple.*

- const [numeric sqrt](#) (const [numeric](#) &x)

*Numeric square root.*

- const [numeric isqrt](#) (const [numeric](#) &x)

*Integer numeric square root.*

- [ex PiEvalf](#) ()

*Floating point evaluation of Archimedes' constant Pi.*

- [ex EulerEvalf](#) ()

*Floating point evaluation of Euler's constant gamma.*

- [ex CatalanEvalf](#) ()

*Floating point evaluation of Catalan's constant.*

- std::ostream & [operator<<](#) (std::ostream &os, const [\\_numeric\\_digits](#) &e)

- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([numeric](#))

- const [numeric pow](#) (const [numeric](#) &x, const [numeric](#) &y)

- const [numeric inverse](#) (const [numeric](#) &x)

- [numeric step](#) (const [numeric](#) &x)

- int [csgn](#) (const [numeric](#) &x)

- bool [is\\_zero](#) (const [numeric](#) &x)

- bool [is\\_positive](#) (const [numeric](#) &x)

- bool [is\\_negative](#) (const [numeric](#) &x)

- bool [is\\_integer](#) (const [numeric](#) &x)

- bool [is\\_pos\\_integer](#) (const [numeric](#) &x)

- bool [is\\_nonneg\\_integer](#) (const [numeric](#) &x)

- bool [is\\_even](#) (const [numeric](#) &x)

- bool [is\\_odd](#) (const [numeric](#) &x)

- bool [is\\_prime](#) (const [numeric](#) &x)

- bool [is\\_rational](#) (const [numeric](#) &x)

- bool [is\\_real](#) (const [numeric](#) &x)

- bool [is\\_cinteger](#) (const [numeric](#) &x)

- bool [is\\_crational](#) (const [numeric](#) &x)



- `int to_int (const numeric &x)`
- `long to_long (const numeric &x)`
- `double to_double (const numeric &x)`
- `const numeric real (const numeric &x)`
- `const numeric imag (const numeric &x)`
- `const numeric numer (const numeric &x)`
- `const numeric denom (const numeric &x)`
- `static const ex exadd (const ex &lh, const ex &rh)`  
*Used internally by `operator+()` to add two ex objects.*
- `static const ex exmul (const ex &lh, const ex &rh)`  
*Used internally by `operator*()` to multiply two ex objects.*
- `static const ex exminus (const ex &lh)`  
*Used internally by `operator-()` and friends to change the sign of an argument.*
- `const ex operator+ (const ex &lh, const ex &rh)`
- `const ex operator- (const ex &lh, const ex &rh)`
- `const ex operator* (const ex &lh, const ex &rh)`
- `const ex operator/ (const ex &lh, const ex &rh)`
- `const numeric operator+ (const numeric &lh, const numeric &rh)`
- `const numeric operator- (const numeric &lh, const numeric &rh)`
- `const numeric operator* (const numeric &lh, const numeric &rh)`
- `const numeric operator/ (const numeric &lh, const numeric &rh)`
- `ex & operator+= (ex &lh, const ex &rh)`
- `ex & operator-= (ex &lh, const ex &rh)`
- `ex & operator*= (ex &lh, const ex &rh)`
- `ex & operator/= (ex &lh, const ex &rh)`
- `numeric & operator+= (numeric &lh, const numeric &rh)`
- `numeric & operator-= (numeric &lh, const numeric &rh)`
- `numeric & operator*= (numeric &lh, const numeric &rh)`
- `numeric & operator/= (numeric &lh, const numeric &rh)`
- `const ex operator+ (const ex &lh)`
- `const ex operator- (const ex &lh)`
- `const numeric operator+ (const numeric &lh)`
- `const numeric operator- (const numeric &lh)`
- `ex & operator++ (ex &rh)`  
*Expression prefix increment.*
- `ex & operator-- (ex &rh)`  
*Expression prefix decrement.*
- `const ex operator++ (ex &lh, int)`  
*Expression postfix increment.*
- `const ex operator-- (ex &lh, int)`  
*Expression postfix decrement.*
- `numeric & operator++ (numeric &rh)`  
*Numeric prefix increment.*
- `numeric & operator-- (numeric &rh)`  
*Numeric prefix decrement.*
- `const numeric operator++ (numeric &lh, int)`  
*Numeric postfix increment.*
- `const numeric operator-- (numeric &lh, int)`  
*Numeric postfix decrement.*
- `const relational operator== (const ex &lh, const ex &rh)`
- `const relational operator!= (const ex &lh, const ex &rh)`
- `const relational operator< (const ex &lh, const ex &rh)`
- `const relational operator<= (const ex &lh, const ex &rh)`

- const [relational operator](#)> (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator](#)>= (const [ex](#) &lh, const [ex](#) &rh)
- static int [my\\_ios\\_index](#) ()
- static void [my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static [print\\_context](#) \* [get\\_print\\_context](#) (std::ios\_base &s)
- static void [set\\_print\\_context](#) (std::ios\_base &s, const [print\\_context](#) &c)
- static unsigned [get\\_print\\_options](#) (std::ios\_base &s)
- static void [set\\_print\\_options](#) (std::ostream &s, unsigned options)
- std::ostream & [operator<<](#) (std::ostream &os, const [ex](#) &e)
- std::istream & [operator>>](#) (std::istream &is, [ex](#) &e)
- std::ostream & [dflt](#) (std::ostream &os)
- std::ostream & [latex](#) (std::ostream &os)
- std::ostream & [python](#) (std::ostream &os)
- std::ostream & [python\\_repr](#) (std::ostream &os)
- std::ostream & [tree](#) (std::ostream &os)
- std::ostream & [csrc](#) (std::ostream &os)
- std::ostream & [csrc\\_float](#) (std::ostream &os)
- std::ostream & [csrc\\_double](#) (std::ostream &os)
- std::ostream & [csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [index\\_dimensions](#) (std::ostream &os)
- std::ostream & [no\\_index\\_dimensions](#) (std::ostream &os)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([power](#), [basic](#), print\_func< [print\\_dflt](#) >(&[power::do\\_print\\_dflt](#)). print\_func< [print\\_latex](#) >(&[power::do\\_print\\_latex](#)). print\_func< [print\\_csrc](#) >(&[power::do\\_print\\_csrc](#)). print←\_func< [print\\_python](#) >(&[power::do\\_print\\_python](#)). print\_func< [print\\_python\\_repr](#) >(&[power::do\\_print\\_python\\_repr](#)). print\_func< [print\\_csrc\\_cl\\_N](#) >(&[power::do\\_print\\_csrc\\_cl\\_N](#))) [power](#)
- static void [print\\_sym\\_pow](#) (const [print\\_context](#) &c, const [symbol](#) &x, int [exp](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([power](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([power](#))
- [ex pow](#) (const [ex](#) &b, const [ex](#) &e)  
*Symbolic exponentiation.*
- template<typename T1, typename T2>  
[ex pow](#) (const T1 &b, const T2 &e)
- [ex sqrt](#) (const [ex](#) &a)  
*Square root expression.*
- template<class T>  
bool [is\\_a](#) (const [print\\_context](#) &obj)  
*Check if obj is a T, including base classes.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([pseries](#), [basic](#), print\_func< [print\\_context](#) >(&[pseries::do\\_print](#)). print\_func< [print\\_latex](#) >(&[pseries::do\\_print\\_latex](#)). print\_func< [print\\_tree](#) >(&[pseries::do\\_print\\_tree](#)). print\_func< [print\\_python](#) >(&[pseries::do\\_print\\_python](#)). print\_func< [print\\_python\\_repr](#) >(&[pseries::do\\_print\\_python\\_repr](#))) [pseries](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex series\\_to\\_poly](#) (const [ex](#) &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [is\\_terminating](#) (const [pseries](#) &s)
- template<typename T>  
[return\\_type\\_t](#) [make\\_return\\_type\\_t](#) (const unsigned rl=0)
- template<class Alg, class Ctx, class T, class C>  
void [set\\_print\\_func](#) (void f(const T &, const C &c, unsigned))  
*Add or replace a print method.*
- template<class Alg, class Ctx, class T, class C>  
void [set\\_print\\_func](#) (void (T::\*f)(const C &, unsigned))  
*Add or replace a print method.*

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`relational`, `basic`, `print_func`< `print_context` >(&`relational::do_print`). `print_func`< `print_tree` >(&`relational::do_print_tree`). `print_func`< `print_python_repr` >(&`relational::do_print_python_repr`) `relational`
- `GINAC_BIND_UNARCHIVER` (`relational`)
- static void `print_operator` (const `print_context` &c, `relational::operators` o)
- `GINAC_DECLARE_UNARCHIVER` (`relational`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`symbol`, `basic`, `print_func`< `print_context` >(&`symbol::do_print`). `print_func`< `print_latex` >(&`symbol::do_print_latex`). `print_func`< `print_tree` >(&`symbol::do_print_tree`). `print_func`< `print_python_repr` >(&`symbol::do_print_python_repr`) `symbol`
- static const std::string & `get_default_TeX_name` (const std::string &name)  
*Return default TeX name for symbol.*
- `GINAC_BIND_UNARCHIVER` (`symbol`)
- `GINAC_BIND_UNARCHIVER` (`realsymbol`)
- `GINAC_BIND_UNARCHIVER` (`possymbol`)
- `GINAC_DECLARE_UNARCHIVER` (`symbol`)
- `GINAC_DECLARE_UNARCHIVER` (`realsymbol`)
- `GINAC_DECLARE_UNARCHIVER` (`possymbol`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`symmetry`, `basic`, `print_func`< `print_context` >(&`symmetry::do_print`). `print_func`< `print_tree` >(&`symmetry::do_print_tree`) `symmetry`
- `GINAC_BIND_UNARCHIVER` (`symmetry`)
- static const `symmetry` & `index0` ()
- static const `symmetry` & `index1` ()
- static const `symmetry` & `index2` ()
- static const `symmetry` & `index3` ()
- const `symmetry` & `not_symmetric` ()
- const `symmetry` & `symmetric2` ()
- const `symmetry` & `symmetric3` ()
- const `symmetry` & `symmetric4` ()
- const `symmetry` & `antisymmetric2` ()
- const `symmetry` & `antisymmetric3` ()
- const `symmetry` & `antisymmetric4` ()
- int `canonicalize` (exvector::iterator v, const `symmetry` &`sy`)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static `ex sy` (const `ex` &e, exvector::const\_iterator `first`, exvector::const\_iterator last, bool asymmetric)
- `ex symmetrize` (const `ex` &e, exvector::const\_iterator `first`, exvector::const\_iterator last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- `ex antisymmetrize` (const `ex` &e, exvector::const\_iterator `first`, exvector::const\_iterator last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- `ex symmetrize_cyclic` (const `ex` &e, exvector::const\_iterator `first`, exvector::const\_iterator last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- `GINAC_DECLARE_UNARCHIVER` (`symmetry`)
- `symmetry sy_none` ()
- `symmetry sy_none` (const `symmetry` &c1, const `symmetry` &c2)
- `symmetry sy_none` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3)
- `symmetry sy_none` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3, const `symmetry` &c4)
- `symmetry sy_sy` ()
- `symmetry sy_sy` (const `symmetry` &c1, const `symmetry` &c2)
- `symmetry sy_sy` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3)
- `symmetry sy_sy` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3, const `symmetry` &c4)
- `symmetry sy_anti` ()
- `symmetry sy_anti` (const `symmetry` &c1, const `symmetry` &c2)
- `symmetry sy_anti` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3)
- `symmetry sy_anti` (const `symmetry` &c1, const `symmetry` &c2, const `symmetry` &c3, const `symmetry` &c4)

- [symmetry sy\\_cycl](#) ()
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [ex symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex antisymmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([tensdelta](#), [tensor](#), print\_func< [print\\_dflt](#) >(&[tensdelta::do\\_print](#)), print\_func< [print\\_latex](#) >(&[tensdelta::do\\_print\\_latex](#))) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([tensmetric](#)  
[print\\_func](#)< [print\\_dflt](#) > (&[tensmetric::do\\_print](#)). [print\\_func](#)< [print\\_latex](#) >(&[tensmetric](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensepsilon](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([spinmetric](#))
- [ex delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensepsilon](#))
- unsigned [log2](#) (unsigned n)  
*Integer binary logarithm.*
- const [numeric multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient n!*
- unsigned [rotate\\_left](#) (unsigned n)  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T>  
int [compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [golden\\_ratio\\_hash](#) (uintptr\_t n)  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It>  
int [permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap>  
int [permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)

- `template<class It, class Cmp, class Swap>`  
`void shaker\_sort (It first, It last, Cmp comp, Swap swapit)`
- `template<class It, class Swap>`  
`void cyclic\_permutation (It first, It last, It new_first, Swap swapit)`
- `template<typename T>`  
`std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T>`  
`std::enable_if<!has\_distance< T >::value, T >::type format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const basic\_multi\_iterator< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter\_indv< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_permutation< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle\_prime< T > &v)`  
*Output operator.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (wildcard, basic, print_func< print\_context >(&wildcard::do_print).`  
`print_func< print\_tree >(&wildcard::do_print_tree). print_func< print\_python\_repr >(&wildcard::do_print_python_repr))`  
`wildcard`
- `GINAC_BIND_UNARCHIVER (wildcard)`
- `bool haswild (const ex &x)`  
*Check whether x has a wildcard anywhere as a subexpression.*
- `GINAC_DECLARE_UNARCHIVER (wildcard)`
- `ex wild (unsigned label=0)`  
*Create a wildcard object with the specified label.*

## Variables

- static `unarchive_table_t` `unarch_table_instance`
- `GiNaC::evalm_map_function` `map_evalm`
- `GiNaC::eval_integ_map_function` `map_eval_integ`
- `template<class OPT>`  
`class_info< OPT > * class_info< OPT >::first = nullptr`
- `template<class OPT>`  
`bool class_info< OPT >::parents_identified = false`
- `tensor`
- const `constant` `Pi` ("Pi", `PiEvalf`, "\\pi", `domain::positive`)  
*Pi.*
- const `constant` `Euler` ("Euler", `EulerEvalf`, "\\gamma\_E", `domain::positive`)  
*Euler's constant.*
- const `constant` `Catalan` ("Catalan", `CatalanEvalf`, "G", `domain::positive`)  
*Catalan's constant.*
- static unsigned const `crctab` [256]
- static `library_init` `library_initializer`  
*For construction of flyweights, etc.*
- const `basic` \* `_num0_bp`
- `idx`
- unsigned `force_include_tgamma` = `tgamma_SERIAL::serial`
- unsigned `force_include_zeta1` = `zeta1_SERIAL::serial`
- `template<>` `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T`(`lst`, `basic`, `print_func< print_context >(&lst::do_print)`, `print_func< print_tree >(&lst::do_print_tree)`) `template<>` bool `Is GINAC_BIND_UNARCHIVER`(`lst`)  
*Specialization of `container::info()` for `lst`.*
- const `numeric` `I` = `numeric`(`cln::complex`(`cln::cl_I`(0), `cln::cl_I`(1)))  
*Imaginary unit.*
- `_numeric_digits` `Digits`  
*Accuracy in decimal digits.*
- unsigned `next_print_context_id` = 0  
*Next free ID for `print_context` types.*
- `template<class T, template< class > class CP>`  
`registered_class_info` `structure< T, CP >::reg_info = registered_class_info`(`registered_class_options`(`structure::get_class_name`("basic", `typeid`(`structure< T, CP >()`)))
- const int `version_major` = `GINACLIB_MAJOR_VERSION`
- const int `version_minor` = `GINACLIB_MINOR_VERSION`
- const int `version_micro` = `GINACLIB_MICRO_VERSION`
- const `numeric` \* `_num_120_p`
- const `ex_ex_120` = `ex`(\*`_num_120_p`)
- const `numeric` \* `_num_60_p`
- const `ex_ex_60` = `ex`(\*`_num_60_p`)
- const `numeric` \* `_num_48_p`
- const `ex_ex_48` = `ex`(\*`_num_48_p`)
- const `numeric` \* `_num_30_p`
- const `ex_ex_30` = `ex`(\*`_num_30_p`)
- const `numeric` \* `_num_25_p`
- const `ex_ex_25` = `ex`(\*`_num_25_p`)
- const `numeric` \* `_num_24_p`
- const `ex_ex_24` = `ex`(\*`_num_24_p`)
- const `numeric` \* `_num_20_p`
- const `ex_ex_20` = `ex`(\*`_num_20_p`)
- const `numeric` \* `_num_18_p`

- const `ex_ex_18` = `ex(*_num_18_p)`
- const `numeric *_num_15_p`
- const `ex_ex_15` = `ex(*_num_15_p)`
- const `numeric *_num_12_p`
- const `ex_ex_12` = `ex(*_num_12_p)`
- const `numeric *_num_11_p`
- const `ex_ex_11` = `ex(*_num_11_p)`
- const `numeric *_num_10_p`
- const `ex_ex_10` = `ex(*_num_10_p)`
- const `numeric *_num_9_p`
- const `ex_ex_9` = `ex(*_num_9_p)`
- const `numeric *_num_8_p`
- const `ex_ex_8` = `ex(*_num_8_p)`
- const `numeric *_num_7_p`
- const `ex_ex_7` = `ex(*_num_7_p)`
- const `numeric *_num_6_p`
- const `ex_ex_6` = `ex(*_num_6_p)`
- const `numeric *_num_5_p`
- const `ex_ex_5` = `ex(*_num_5_p)`
- const `numeric *_num_4_p`
- const `ex_ex_4` = `ex(*_num_4_p)`
- const `numeric *_num_3_p`
- const `ex_ex_3` = `ex(*_num_3_p)`
- const `numeric *_num_2_p`
- const `ex_ex_2` = `ex(*_num_2_p)`
- const `numeric *_num_1_p`
- const `ex_ex_1` = `ex(*_num_1_p)`
- const `numeric *_num_1_2_p`
- const `ex_ex_1_2` = `ex(*_num_1_2_p)`
- const `numeric *_num_1_3_p`
- const `ex_ex_1_3` = `ex(*_num_1_3_p)`
- const `numeric *_num_1_4_p`
- const `ex_ex_1_4` = `ex(*_num_1_4_p)`
- const `numeric *_num0_p`
- const `ex_ex0` = `ex(*_num0_p)`
- const `numeric *_num1_4_p`
- const `ex_ex1_4` = `ex(*_num1_4_p)`
- const `numeric *_num1_3_p`
- const `ex_ex1_3` = `ex(*_num1_3_p)`
- const `numeric *_num1_2_p`
- const `ex_ex1_2` = `ex(*_num1_2_p)`
- const `numeric *_num1_p`
- const `ex_ex1` = `ex(*_num1_p)`
- const `numeric *_num2_p`
- const `ex_ex2` = `ex(*_num2_p)`
- const `numeric *_num3_p`
- const `ex_ex3` = `ex(*_num3_p)`
- const `numeric *_num4_p`
- const `ex_ex4` = `ex(*_num4_p)`
- const `numeric *_num5_p`
- const `ex_ex5` = `ex(*_num5_p)`
- const `numeric *_num6_p`
- const `ex_ex6` = `ex(*_num6_p)`
- const `numeric *_num7_p`
- const `ex_ex7` = `ex(*_num7_p)`

- `const numeric * _num8_p`
- `const ex _ex8 = ex(*_num8_p)`
- `const numeric * _num9_p`
- `const ex _ex9 = ex(*_num9_p)`
- `const numeric * _num10_p`
- `const ex _ex10 = ex(*_num10_p)`
- `const numeric * _num11_p`
- `const ex _ex11 = ex(*_num11_p)`
- `const numeric * _num12_p`
- `const ex _ex12 = ex(*_num12_p)`
- `const numeric * _num15_p`
- `const ex _ex15 = ex(*_num15_p)`
- `const numeric * _num18_p`
- `const ex _ex18 = ex(*_num18_p)`
- `const numeric * _num20_p`
- `const ex _ex20 = ex(*_num20_p)`
- `const numeric * _num24_p`
- `const ex _ex24 = ex(*_num24_p)`
- `const numeric * _num25_p`
- `const ex _ex25 = ex(*_num25_p)`
- `const numeric * _num30_p`
- `const ex _ex30 = ex(*_num30_p)`
- `const numeric * _num48_p`
- `const ex _ex48 = ex(*_num48_p)`
- `const numeric * _num60_p`
- `const ex _ex60 = ex(*_num60_p)`
- `const numeric * _num120_p`
- `const ex _ex120 = ex(*_num120_p)`

## 7.1.1 Typedef Documentation

### 7.1.1.1 `archive_node_id`

```
typedef unsigned GiNaC::archive_node_id
```

Numerical ID value to refer to an [archive\\_node](#).

### 7.1.1.2 `archive_atom`

```
typedef unsigned GiNaC::archive_atom
```

Numerical ID value to refer to a string.

### 7.1.1.3 `synthesize_func`

```
typedef basic *(* GiNaC::synthesize_func) ()
```



#### 7.1.1.4 unarchive\_map\_t

```
typedef std::map<std::string, synthesize_func> GiNaC::unarchive_map_t
```

#### 7.1.1.5 exvector

```
typedef std::vector<ex> GiNaC::exvector
```

#### 7.1.1.6 exset

```
typedef std::set<ex, ex_is_less> GiNaC::exset
```

#### 7.1.1.7 exmap

```
typedef std::map<ex, ex, ex_is_less> GiNaC::exmap
```

#### 7.1.1.8 evalffunctype

```
typedef ex(* GiNaC::evalffunctype) ()
```

#### 7.1.1.9 FUNCP\_1P

```
typedef double(* GiNaC::FUNCP_1P) (double)
```

Function pointer with one function parameter.

#### 7.1.1.10 FUNCP\_2P

```
typedef double(* GiNaC::FUNCP_2P) (double, double)
```

Function pointer with two function parameters.

#### 7.1.1.11 FUNCP\_CUBA

```
typedef void(* GiNaC::FUNCP_CUBA) (const int *, const double[], const int *, double[])
```

Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).

#### 7.1.1.12 epvector

```
typedef std::vector<expair> GiNaC::epvector
```

expair-vector

#### 7.1.1.13 epp

```
typedef epvector::iterator GiNaC::epp
```

expair-vector pointer

#### 7.1.1.14 exprseq

```
typedef container<std::vector> GiNaC::exprseq
```

#### 7.1.1.15 paramset

```
typedef std::multiset<unsigned> GiNaC::paramset
```

#### 7.1.1.16 eval\_funcp

```
typedef ex(* GiNaC::eval_funcp) ()
```

#### 7.1.1.17 evalf\_funcp

```
typedef ex(* GiNaC::evalf_funcp) ()
```

#### 7.1.1.18 conjugate\_funcp

```
typedef ex(* GiNaC::conjugate_funcp) ()
```

#### 7.1.1.19 real\_part\_funcp

```
typedef ex(* GiNaC::real_part_funcp) ()
```

#### 7.1.1.20 imag\_part\_funcp

```
typedef ex(* GiNaC::imag_part_funcp) ()
```

#### 7.1.1.21 expand\_funcp

```
typedef ex(* GiNaC::expand_funcp) ()
```

#### 7.1.1.22 derivative\_funcp

```
typedef ex(* GiNaC::derivative_funcp) ()
```

#### 7.1.1.23 `expl_derivative_funcp`

```
typedef ex(* GiNaC::expl_derivative_funcp) ()
```

#### 7.1.1.24 `power_funcp`

```
typedef ex(* GiNaC::power_funcp) ()
```

#### 7.1.1.25 `series_funcp`

```
typedef ex(* GiNaC::series_funcp) ()
```

#### 7.1.1.26 `print_funcp`

```
typedef void(* GiNaC::print_funcp) ()
```

#### 7.1.1.27 `info_funcp`

```
typedef bool(* GiNaC::info_funcp) ()
```

#### 7.1.1.28 `eval_funcp_1`

```
typedef ex(* GiNaC::eval_funcp_1) (const ex &)
```

#### 7.1.1.29 `evalf_funcp_1`

```
typedef ex(* GiNaC::evalf_funcp_1) (const ex &)
```

#### 7.1.1.30 `conjugate_funcp_1`

```
typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)
```

#### 7.1.1.31 `real_part_funcp_1`

```
typedef ex(* GiNaC::real_part_funcp_1) (const ex &)
```

#### 7.1.1.32 `imag_part_funcp_1`

```
typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)
```

#### 7.1.1.33 expand\_funcp\_1

```
typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)
```

#### 7.1.1.34 derivative\_funcp\_1

```
typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)
```

#### 7.1.1.35 expl\_derivative\_funcp\_1

```
typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)
```

#### 7.1.1.36 power\_funcp\_1

```
typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)
```

#### 7.1.1.37 series\_funcp\_1

```
typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)
```

#### 7.1.1.38 print\_funcp\_1

```
typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)
```

#### 7.1.1.39 info\_funcp\_1

```
typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)
```

#### 7.1.1.40 eval\_funcp\_2

```
typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)
```

#### 7.1.1.41 evalf\_funcp\_2

```
typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)
```

#### 7.1.1.42 conjugate\_funcp\_2

```
typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)
```

#### 7.1.1.43 real\_part\_funcp\_2

```
typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)
```

#### 7.1.1.44 imag\_part\_funcp\_2

```
typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)
```

#### 7.1.1.45 expand\_funcp\_2

```
typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)
```

#### 7.1.1.46 derivative\_funcp\_2

```
typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)
```

#### 7.1.1.47 expl\_derivative\_funcp\_2

```
typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)
```

#### 7.1.1.48 power\_funcp\_2

```
typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)
```

#### 7.1.1.49 series\_funcp\_2

```
typedef ex(* GiNaC::series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
```

#### 7.1.1.50 print\_funcp\_2

```
typedef void(* GiNaC::print_funcp_2) (const ex &, const ex &, const print_context &)
```

#### 7.1.1.51 info\_funcp\_2

```
typedef bool(* GiNaC::info_funcp_2) (const ex &, const ex &, unsigned)
```

#### 7.1.1.52 eval\_funcp\_3

```
typedef ex(* GiNaC::eval_funcp_3) (const ex &, const ex &, const ex &)
```

#### 7.1.1.53 evalf\_funcp\_3

```
typedef ex(* GiNaC::evalf_funcp_3) (const ex &, const ex &, const ex &)
```

#### 7.1.1.54 conjugate\_funcp\_3

```
typedef ex(* GiNaC::conjugate_funcp_3) (const ex &, const ex &, const ex &)
```

#### 7.1.1.55 real\_part\_funcp\_3

```
typedef ex(* GiNaC::real_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 7.1.1.56 imag\_part\_funcp\_3

```
typedef ex(* GiNaC::imag_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 7.1.1.57 expand\_funcp\_3

```
typedef ex(* GiNaC::expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.58 derivative\_funcp\_3

```
typedef ex(* GiNaC::derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.59 expl\_derivative\_funcp\_3

```
typedef ex(* GiNaC::expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol  
&)
```

#### 7.1.1.60 power\_funcp\_3

```
typedef ex(* GiNaC::power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.61 series\_funcp\_3

```
typedef ex(* GiNaC::series_funcp_3) (const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 7.1.1.62 print\_funcp\_3

```
typedef void(* GiNaC::print_funcp_3) (const ex &, const ex &, const ex &, const print_context  
&)
```

#### 7.1.1.63 info\_funcp\_3

```
typedef bool(* GiNaC::info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.64 eval\_funcp\_4

```
typedef ex(* GiNaC::eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.65 evalf\_funcp\_4

```
typedef ex(* GiNaC::evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.66 conjugate\_funcp\_4

```
typedef ex(* GiNaC::conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.67 real\_part\_funcp\_4

```
typedef ex(* GiNaC::real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.68 imag\_part\_funcp\_4

```
typedef ex(* GiNaC::imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.69 expand\_funcp\_4

```
typedef ex(* GiNaC::expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.70 derivative\_funcp\_4

```
typedef ex(* GiNaC::derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.71 expl\_derivative\_funcp\_4

```
typedef ex(* GiNaC::expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &,  
const symbol &)
```

#### 7.1.1.72 power\_funcp\_4

```
typedef ex(* GiNaC::power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex  
&)
```

#### 7.1.1.73 series\_funcp\_4

```
typedef ex(* GiNaC::series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 7.1.1.74 print\_funcp\_4

```
typedef void(* GiNaC::print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 7.1.1.75 info\_funcp\_4

```
typedef bool(* GiNaC::info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.76 eval\_funcp\_5

```
typedef ex(* GiNaC::eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.77 evalf\_funcp\_5

```
typedef ex(* GiNaC::evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.78 conjugate\_funcp\_5

```
typedef ex(* GiNaC::conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.79 real\_part\_funcp\_5

```
typedef ex(* GiNaC::real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.80 imag\_part\_funcp\_5

```
typedef ex(* GiNaC::imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.81 expand\_funcp\_5

```
typedef ex(* GiNaC::expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```



#### 7.1.1.82 derivative\_funcp\_5

```
typedef ex(* GiNaC::derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.83 expl\_derivative\_funcp\_5

```
typedef ex(* GiNaC::expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 7.1.1.84 power\_funcp\_5

```
typedef ex(* GiNaC::power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.85 series\_funcp\_5

```
typedef ex(* GiNaC::series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 7.1.1.86 print\_funcp\_5

```
typedef void(* GiNaC::print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 7.1.1.87 info\_funcp\_5

```
typedef bool(* GiNaC::info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.88 eval\_funcp\_6

```
typedef ex(* GiNaC::eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.89 evalf\_funcp\_6

```
typedef ex(* GiNaC::evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.90 conjugate\_funcp\_6

```
typedef ex(* GiNaC::conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.91 real\_part\_funcp\_6**

```
typedef ex(* GiNaC::real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**7.1.1.92 imag\_part\_funcp\_6**

```
typedef ex(* GiNaC::imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**7.1.1.93 expand\_funcp\_6**

```
typedef ex(* GiNaC::expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, unsigned)
```

**7.1.1.94 derivative\_funcp\_6**

```
typedef ex(* GiNaC::derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

**7.1.1.95 expl\_derivative\_funcp\_6**

```
typedef ex(* GiNaC::expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const symbol &)
```

**7.1.1.96 power\_funcp\_6**

```
typedef ex(* GiNaC::power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &)
```

**7.1.1.97 series\_funcp\_6**

```
typedef ex(* GiNaC::series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const relational &, int, unsigned)
```

**7.1.1.98 print\_funcp\_6**

```
typedef void(* GiNaC::print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const print_context &)
```

**7.1.1.99 info\_funcp\_6**

```
typedef bool(* GiNaC::info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, unsigned)
```

#### 7.1.1.100 eval\_funcp\_7

```
typedef ex(* GiNaC::eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

#### 7.1.1.101 evalf\_funcp\_7

```
typedef ex(* GiNaC::evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &)
```

#### 7.1.1.102 conjugate\_funcp\_7

```
typedef ex(* GiNaC::conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 7.1.1.103 real\_part\_funcp\_7

```
typedef ex(* GiNaC::real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 7.1.1.104 imag\_part\_funcp\_7

```
typedef ex(* GiNaC::imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 7.1.1.105 expand\_funcp\_7

```
typedef ex(* GiNaC::expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

#### 7.1.1.106 derivative\_funcp\_7

```
typedef ex(* GiNaC::derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.107 expl\_derivative\_funcp\_7

```
typedef ex(* GiNaC::expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const symbol &)
```

#### 7.1.1.108 power\_funcp\_7

```
typedef ex(* GiNaC::power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

**7.1.1.109 series\_funcp\_7**

```
typedef ex(* GiNaC::series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const relational &, int, unsigned)
```

**7.1.1.110 print\_funcp\_7**

```
typedef void(* GiNaC::print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const print_context &)
```

**7.1.1.111 info\_funcp\_7**

```
typedef bool(* GiNaC::info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

**7.1.1.112 eval\_funcp\_8**

```
typedef ex(* GiNaC::eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &)
```

**7.1.1.113 evalf\_funcp\_8**

```
typedef ex(* GiNaC::evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

**7.1.1.114 conjugate\_funcp\_8**

```
typedef ex(* GiNaC::conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

**7.1.1.115 real\_part\_funcp\_8**

```
typedef ex(* GiNaC::real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

**7.1.1.116 imag\_part\_funcp\_8**

```
typedef ex(* GiNaC::imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

**7.1.1.117 expand\_funcp\_8**

```
typedef ex(* GiNaC::expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.118 derivative\_funcp\_8

```
typedef ex(* GiNaC::derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.119 expl\_derivative\_funcp\_8

```
typedef ex(* GiNaC::expl_derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 7.1.1.120 power\_funcp\_8

```
typedef ex(* GiNaC::power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.121 series\_funcp\_8

```
typedef ex(* GiNaC::series_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 7.1.1.122 print\_funcp\_8

```
typedef void(* GiNaC::print_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 7.1.1.123 info\_funcp\_8

```
typedef bool(* GiNaC::info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.124 eval\_funcp\_9

```
typedef ex(* GiNaC::eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.125 evalf\_funcp\_9

```
typedef ex(* GiNaC::evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.126 conjugate\_funcp\_9

```
typedef ex(* GiNaC::conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.127 real\_part\_funcp\_9**

```
typedef ex(* GiNaC::real_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.128 imag\_part\_funcp\_9**

```
typedef ex(* GiNaC::imag_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.129 expand\_funcp\_9**

```
typedef ex(* GiNaC::expand_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**7.1.1.130 derivative\_funcp\_9**

```
typedef ex(* GiNaC::derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**7.1.1.131 expl\_derivative\_funcp\_9**

```
typedef ex(* GiNaC::expl_derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**7.1.1.132 power\_funcp\_9**

```
typedef ex(* GiNaC::power_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.133 series\_funcp\_9**

```
typedef ex(* GiNaC::series_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

**7.1.1.134 print\_funcp\_9**

```
typedef void(* GiNaC::print_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**7.1.1.135 info\_funcp\_9**

```
typedef bool(* GiNaC::info_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.136 eval\_funcp\_10

```
typedef ex(* GiNaC::eval_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.137 evalf\_funcp\_10

```
typedef ex(* GiNaC::evalf_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.138 conjugate\_funcp\_10

```
typedef ex(* GiNaC::conjugate_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.139 real\_part\_funcp\_10

```
typedef ex(* GiNaC::real_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.140 imag\_part\_funcp\_10

```
typedef ex(* GiNaC::imag_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.141 expand\_funcp\_10

```
typedef ex(* GiNaC::expand_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.142 derivative\_funcp\_10

```
typedef ex(* GiNaC::derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.143 expl\_derivative\_funcp\_10

```
typedef ex(* GiNaC::expl_derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 7.1.1.144 power\_funcp\_10

```
typedef ex(* GiNaC::power_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.145 series\_funcp\_10**

```
typedef ex(* GiNaC::series_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int,
unsigned)
```

**7.1.1.146 print\_funcp\_10**

```
typedef void(* GiNaC::print_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**7.1.1.147 info\_funcp\_10**

```
typedef bool(* GiNaC::info_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**7.1.1.148 eval\_funcp\_11**

```
typedef ex(* GiNaC::eval_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.149 evalf\_funcp\_11**

```
typedef ex(* GiNaC::evalf_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.150 conjugate\_funcp\_11**

```
typedef ex(* GiNaC::conjugate_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.151 real\_part\_funcp\_11**

```
typedef ex(* GiNaC::real_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.152 imag\_part\_funcp\_11**

```
typedef ex(* GiNaC::imag_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.153 expand\_funcp\_11**

```
typedef ex(* GiNaC::expand_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```



#### 7.1.1.154 derivative\_funcp\_11

```
typedef ex(* GiNaC::derivative_funcp_11) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.155 expl\_derivative\_funcp\_11

```
typedef ex(* GiNaC::expl_derivative_funcp_11) (const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
symbol &)
```

#### 7.1.1.156 power\_funcp\_11

```
typedef ex(* GiNaC::power_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.157 series\_funcp\_11

```
typedef ex(* GiNaC::series_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &,
int, unsigned)
```

#### 7.1.1.158 print\_funcp\_11

```
typedef void(* GiNaC::print_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context
&)
```

#### 7.1.1.159 info\_funcp\_11

```
typedef bool(* GiNaC::info_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 7.1.1.160 eval\_funcp\_12

```
typedef ex(* GiNaC::eval_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 7.1.1.161 evalf\_funcp\_12

```
typedef ex(* GiNaC::evalf_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.162 conjugate\_funcp\_12**

```
typedef ex(* GiNaC::conjugate_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.163 real\_part\_funcp\_12**

```
typedef ex(* GiNaC::real_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.164 imag\_part\_funcp\_12**

```
typedef ex(* GiNaC::imag_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**7.1.1.165 expand\_funcp\_12**

```
typedef ex(* GiNaC::expand_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
unsigned)
```

**7.1.1.166 derivative\_funcp\_12**

```
typedef ex(* GiNaC::derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, unsigned)
```

**7.1.1.167 expl\_derivative\_funcp\_12**

```
typedef ex(* GiNaC::expl_derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const symbol &)
```

**7.1.1.168 power\_funcp\_12**

```
typedef ex(* GiNaC::power_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

**7.1.1.169 series\_funcp\_12**

```
typedef ex(* GiNaC::series_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
relational &, int, unsigned)
```

#### 7.1.1.170 print\_funcp\_12

```
typedef void(* GiNaC::print_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const print_context &)
```

#### 7.1.1.171 info\_funcp\_12

```
typedef bool(* GiNaC::info_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
unsigned)
```

#### 7.1.1.172 eval\_funcp\_13

```
typedef ex(* GiNaC::eval_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 7.1.1.173 evalf\_funcp\_13

```
typedef ex(* GiNaC::evalf_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 7.1.1.174 conjugate\_funcp\_13

```
typedef ex(* GiNaC::conjugate_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

#### 7.1.1.175 real\_part\_funcp\_13

```
typedef ex(* GiNaC::real_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

#### 7.1.1.176 imag\_part\_funcp\_13

```
typedef ex(* GiNaC::imag_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

#### 7.1.1.177 expand\_funcp\_13

```
typedef ex(* GiNaC::expand_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, unsigned)
```

**7.1.1.178 derivative\_funcp\_13**

```
typedef ex(* GiNaC::derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
&, const ex &, unsigned)
```

**7.1.1.179 expl\_derivative\_funcp\_13**

```
typedef ex(* GiNaC::expl_derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
&, const ex &, const symbol &)
```

**7.1.1.180 power\_funcp\_13**

```
typedef ex(* GiNaC::power_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
ex &, const ex &)
```

**7.1.1.181 series\_funcp\_13**

```
typedef ex(* GiNaC::series_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
ex &, const relational &, int, unsigned)
```

**7.1.1.182 print\_funcp\_13**

```
typedef void(* GiNaC::print_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const print_context &)
```

**7.1.1.183 info\_funcp\_13**

```
typedef bool(* GiNaC::info_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
ex &, unsigned)
```

**7.1.1.184 eval\_funcp\_14**

```
typedef ex(* GiNaC::eval_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
ex &, const ex &)
```

**7.1.1.185 evalf\_funcp\_14**

```
typedef ex(* GiNaC::evalf_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
ex &, const ex &)
```

**7.1.1.186 conjugate\_funcp\_14**

```
typedef ex(* GiNaC::conjugate_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

**7.1.1.187 real\_part\_funcp\_14**

```
typedef ex(* GiNaC::real_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

**7.1.1.188 imag\_part\_funcp\_14**

```
typedef ex(* GiNaC::imag_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

**7.1.1.189 expand\_funcp\_14**

```
typedef ex(* GiNaC::expand_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

**7.1.1.190 derivative\_funcp\_14**

```
typedef ex(* GiNaC::derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

**7.1.1.191 expl\_derivative\_funcp\_14**

```
typedef ex(* GiNaC::expl_derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const symbol &)
```

**7.1.1.192 power\_funcp\_14**

```
typedef ex(* GiNaC::power_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

**7.1.1.193 series\_funcp\_14**

```
typedef ex(* GiNaC::series_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const relational &, int, unsigned)
```

**7.1.1.194 print\_funcp\_14**

```
typedef void(* GiNaC::print_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const print_context &)
```

**7.1.1.195 info\_funcp\_14**

```
typedef bool(* GiNaC::info_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

**7.1.1.196 eval\_funcp\_exvector**

```
typedef ex(* GiNaC::eval_funcp_exvector) (const exvector &)
```

**7.1.1.197 evalf\_funcp\_exvector**

```
typedef ex(* GiNaC::evalf_funcp_exvector) (const exvector &)
```

**7.1.1.198 conjugate\_funcp\_exvector**

```
typedef ex(* GiNaC::conjugate_funcp_exvector) (const exvector &)
```

**7.1.1.199 real\_part\_funcp\_exvector**

```
typedef ex(* GiNaC::real_part_funcp_exvector) (const exvector &)
```

**7.1.1.200 imag\_part\_funcp\_exvector**

```
typedef ex(* GiNaC::imag_part_funcp_exvector) (const exvector &)
```

**7.1.1.201 expand\_funcp\_exvector**

```
typedef ex(* GiNaC::expand_funcp_exvector) (const exvector &, unsigned)
```

**7.1.1.202 derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::derivative_funcp_exvector) (const exvector &, unsigned)
```

**7.1.1.203 expl\_derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::expl_derivative_funcp_exvector) (const exvector &, const symbol &)
```

**7.1.1.204 power\_funcp\_exvector**

```
typedef ex(* GiNaC::power_funcp_exvector) (const exvector &, const ex &)
```

**7.1.1.205 series\_funcp\_exvector**

```
typedef ex(* GiNaC::series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
```

**7.1.1.206 print\_funcp\_exvector**

```
typedef void(* GiNaC::print_funcp_exvector) (const exvector &, const print_context &)
```

**7.1.1.207 info\_funcp\_exvector**

```
typedef bool(* GiNaC::info_funcp_exvector) (const exvector &, unsigned)
```

**7.1.1.208 exhashmap**

```
template<typename T, class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class  
Allocator = std::allocator<std::pair<const ex, T>>>  
using GiNaC::exhashmap = std::unordered_map<ex, T, Hash, KeyEqual, Allocator>
```

**7.1.1.209 spmap**

```
typedef std::map<spmapkey, ex> GiNaC::spmap
```

**7.1.1.210 lookup\_map**

```
typedef map<error_and_integral, ex, error_and_integral_is_less> GiNaC::lookup_map
```

**7.1.1.211 lst**

```
typedef container< std::list > GiNaC::lst
```

**7.1.1.212 uintvector**

```
typedef std::vector<std::size_t> GiNaC::uintvector
```

**7.1.1.213 unsignedvector**

```
typedef std::vector<unsigned> GiNaC::unsignedvector
```

#### 7.1.1.214 `exvectorvector`

```
typedef std::vector<exvector> GiNaC::exvectorvector
```

#### 7.1.1.215 `sym_desc_vec`

```
typedef std::vector<sym_desc> GiNaC::sym_desc_vec
```

#### 7.1.1.216 `digits_changed_callback`

```
typedef void(* GiNaC::digits_changed_callback) (long)
```

Function pointer to implement callbacks in the case 'Digits' gets changed.

Main purpose of such callbacks is to adjust look-up tables of certain functions to the new precision. Parameter contains the signed difference between new Digits and old Digits.

#### 7.1.1.217 `print_context_class_info`

```
typedef class_info<print_context_options> GiNaC::print_context_class_info
```

#### 7.1.1.218 `registered_class_info`

```
typedef class_info<registered_class_options> GiNaC::registered_class_info
```

### 7.1.2 Enumeration Type Documentation

#### 7.1.2.1 anonymous enum

anonymous enum

##### Enumerator

callback_registered	
---------------------	--

### 7.1.3 Function Documentation

#### 7.1.3.1 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [1/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    add ,
    expairseq ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )
```

References [GiNaC::add::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::add::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), and [GiNaC::basic::do\\_print\\_tree\(\)](#).

Referenced by [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [print\\_func< print\\_dflt >\(\)](#), [print\\_func< print\\_dflt >\(\)](#), and [print\\_func< print\\_dflt >\(\)](#).



### 7.1.3.2 GINAC\_BIND\_UNARCHIVER() [1/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (  
    add )
```

### 7.1.3.3 GINAC\_DECLARE\_UNARCHIVER() [1/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (  
    add )
```

### 7.1.3.4 write\_unsigned()

```
void GiNaC::write_unsigned (  
    std::ostream & os,  
    unsigned val) [static]
```

Write unsigned integer quantity to stream.

Referenced by [GiNaC::archive::operator<<](#), and [GiNaC::archive\\_node::operator<<](#).

### 7.1.3.5 read\_unsigned()

```
unsigned GiNaC::read_unsigned (  
    std::istream & is) [static]
```

Read unsigned integer quantity from stream.

Referenced by [GiNaC::archive::operator>>](#), and [GiNaC::archive\\_node::operator>>](#).

### 7.1.3.6 operator<<() [1/16]

```
std::ostream & GiNaC::operator<< (  
    std::ostream & os,  
    const archive_node & n)
```

Write [archive\\_node](#) to binary data stream.

### 7.1.3.7 operator<<() [2/16]

```
std::ostream & GiNaC::operator<< (  
    std::ostream & os,  
    const archive & ar)
```

Write archive to binary data stream.

**7.1.3.8 operator>>() [1/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    archive_node & n)
```

Read [archive\\_node](#) from binary data stream.

**7.1.3.9 operator>>() [2/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    archive & ar)
```

Read archive from binary data stream.

**7.1.3.10 find\_factory\_fcn()**

```
synthesize_func GiNaC::find_factory_fcn (
    const std::string & name) [static]
```

References [GiNaC::unarchive\\_table\\_t::find\(\)](#).

Referenced by [GiNaC::archive\\_node::unarchive\(\)](#).

**7.1.3.11 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [2/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic ,
    void ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree. print_func< print_python_repr > &::do_print_python_repr )
```

basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.

References [GiNaC::basic::do\\_print\(\)](#), [GiNaC::basic::do\\_print\\_python\\_repr\(\)](#), and [GiNaC::basic::do\\_print\\_tree\(\)](#).

**7.1.3.12 is\_a()** [1/3]

```
template<class T>
bool GiNaC::is_a (
    const basic & obj) [inline]
```

Check if obj is a T, including base classes.

Referenced by [abs\\_power\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [base\\_and\\_index\(\)](#), [beta\\_series\(\)](#), [canonicalize\\_clifford\(\)](#), [GiNaC::clifford::clifford\(\)](#), [clifford\\_max\\_label\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_prime\(\)](#), [clifford\\_star\\_bar\(\)](#), [clifford\\_to\\_lst\(\)](#), [clifford\\_unit\(\)](#), [GiNaC::basic::collect\(\)](#), [collect\\_symbols\(\)](#), [color\\_d\(\)](#), [color\\_f\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::cliffordunit::contract\\_with\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [GiNaC::tensdelta::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [GiNaC::tensmetric::contract\\_with\(\)](#), [convert\\_H\\_to\\_Li\(\)](#), [csgn\\_power\(\)](#), [delta\\_tensor\(\)](#), [GiNaC::power::derivative\(\)](#), [dirac\\_gamma\(\)](#), [dirac\\_trace\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [epsilon\\_tensor\(\)](#), [epsilon\\_tensor\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::color::eval\\_ncmul\(\)](#), [GiNaC::Gt::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [expand\\_dummy\\_sum\(\)](#), [find\\_common\\_factor\(\)](#), [fsolve\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [get\\_clifford\\_comp\(\)](#), [get\\_dim\\_uint\(\)](#), [get\\_first\\_symbol\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GINAC\\_BIND\\_UNARCHIVER\(\)](#), [GiNaC::Gt::Gt\(\)](#), [GiNaC::Gt::Gt\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [H\\_evalf\(\)](#), [H\\_print\\_latex\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), [hasindex\(\)](#), [haswild\(\)](#), [GiNaC::symbolset::insert\\_symbols\(\)](#), [GiNaC::integral::integral\(\)](#), [GiNaC::ex::is\\_a\(\)](#), [is\\_dirac\\_slash\(\)](#), [is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::fderivative::is\\_equal\\_same\\_type\(\)](#), [GiNaC::function::is\\_equal\\_same\\_type\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::symbol::is\\_equal\\_same\\_type\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [lcmcoeff\(\)](#), [Li\\_deriv\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [Li\\_print\\_latex\(\)](#), [Li\\_series\(\)](#), [log\\_series\(\)](#), [lorentz\\_eps\(\)](#), [lorentz\\_g\(\)](#), [lst\\_to\\_clifford\(\)](#), [lst\\_to\\_matrix\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::expairseq::map\(\)](#), [GiNaC::clifford::match\\_same\\_type\(\)](#), [GiNaC::color::match\\_same\\_type\(\)](#), [GiNaC::fderivative::match\\_same\\_type\(\)](#), [GiNaC::function::match\\_same\\_type\(\)](#), [GiNaC::idx::match\\_same\\_type\(\)](#), [GiNaC::spinidx::match\\_same\\_type\(\)](#), [GiNaC::varidx::match\\_same\\_type\(\)](#), [metric\\_tensor\(\)](#), [minimal\\_dim\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::ex\\_base\\_is\\_less::operator\(\)](#), [GiNaC::is\\_not\\_a\\_clifford::operator\(\)](#), [GiNaC::spmapkey::operator<\(\)](#), [GiNaC::spmapkey::operator==\(\)](#), [Order\\_series\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [print\\_real\\_number\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [remove\\_dirac\\_ONE\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices\(\)](#), [GiNaC::indexed::return\\_type\(\)](#), [S\\_evalf\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::Gt::setArgs\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [GiNaC::indexed::simplify\\_indexed\\_product\(\)](#), [spinor\\_metric\(\)](#), [GiNaC::spmapkey::spmapkey\(\)](#), [sqrfree\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [subsvalue\(\)](#), [tan\\_series\(\)](#), [tanh\\_series\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), [GiNaC::Gt::to\\_numeric\(\)](#), [GiNaC::indexed::validate\(\)](#), [zeta1\\_print\\_latex\(\)](#), and [zeta2\\_print\\_latex\(\)](#).

**7.1.3.13 is\_exactly\_a()** [1/2]

```
template<class T>
bool GiNaC::is_exactly_a (
    const basic & obj) [inline]
```

Check if obj is a T, not including base classes.

Referenced by [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_expand\(\)](#), [acos\\_conjugate\(\)](#), [acos\\_evalf\(\)](#), [acosh\\_conjugate\(\)](#), [acosh\\_evalf\(\)](#), [adaptivesimpson\(\)](#), [GiNaC::symmetry::add\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#),

[asin\\_conjugate\(\)](#), [asin\\_evalf\(\)](#), [asinh\\_conjugate\(\)](#), [asinh\\_evalf\(\)](#), [atan2\\_evalf\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_evalf\(\)](#), [atanh\\_conjugate\(\)](#), [atanh\\_evalf\(\)](#), [beta\\_evalf\(\)](#), [binomial\\_eval\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::symmetry::canonicalize](#), [canonicalize\\_clifford\(\)](#), [GiNaC::Gt\\_detail::TransformExpressionWithCache](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [collect\\_common\\_factors\(\)](#), [collect\\_symbols\(\)](#), [color\\_trace\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\\_evalf\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [decomp\\_rational\(\)](#), [GiNaC::power::degree\(\)](#), [dirac\\_trace\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::power::evalf\(\)](#), [GiNaC::power::evalm\(\)](#), [exp\\_evalf\(\)](#), [exp\\_expand\(\)](#), [GiNaC::expair::expair\(\)](#), [GiNaC::expairseq::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [factorial\\_eval\(\)](#), [factorial\\_print\\_dflt\\_latex\(\)](#), [find\\_common\\_factor\(\)](#), [find\\_variant\\_indices\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [get\\_first\\_symbol\(\)](#), [GiNaC::numeric::has\(\)](#), [heur\\_gcd\\_z\(\)](#), [idx\\_symmetrization\(\)](#), [imag\\_part\\_evalf\(\)](#), [GiNaC::Gt\\_detail::TransformExpressionWithCache< Type >::impl\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [GiNaC::expairseq::is\\_canonical\(\)](#), [GiNaC::expair::is\\_canonical\\_numeric\(\)](#), [GiNaC::idx::is\\_dim\\_numeric\(\)](#), [GiNaC::idx::is\\_dim\\_symbolic\(\)](#), [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#), [GiNaC::constant::is\\_equal\\_same\\_type\(\)](#), [GiNaC::numeric::is\\_equal\\_same\\_type\(\)](#), [GiNaC::ex::is\\_exactly\\_a](#), [GiNaC::idx::is\\_numeric\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), [is\\_the\\_function\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [GiNaC::power::ldegree\(\)](#), [lgamma\\_conjugate\(\)](#), [lgamma\\_evalf\(\)](#), [Li2\\_conjugate\(\)](#), [Li2\\_evalf\(\)](#), [log\\_conjugate\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [lsolve\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::matrix::match\\_same\\_type\(\)](#), [GiNaC::relational::match\\_same\\_type\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [minimal\\_dim\(\)](#), [multiply\\_lcm\(\)](#), [number\\_of\\_type\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::sy\\_is\\_less::operator\(\)](#), [GiNaC::sy\\_swap::operator\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [Order\\_eval\(\)](#), [Order\\_power\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::matrix::pow\(\)](#), [prem\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [product\\_to\\_exvector\(\)](#), [psi1\\_evalf\(\)](#), [psi2\\_evalf\(\)](#), [quo\(\)](#), [real\\_part\\_evalf\(\)](#), [rem\(\)](#), [rename\\_dummy\\_indices\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices](#), [GiNaC::add::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [GiNaC::indexed::simplify\\_indexed\\_product](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [sprem\(\)](#), [sqrfree\(\)](#), [sr\\_gcd\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tgamma\\_evalf\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [tryfactsubs\(\)](#), [GiNaC::indexed::validate\(\)](#), [zeta1\\_deriv\(\)](#), [zeta1\\_eval\(\)](#), [zeta1\\_evalf\(\)](#), [zeta2\\_deriv\(\)](#), [zeta2\\_eval\(\)](#), and [zeta2\\_evalf\(\)](#).

#### 7.1.3.14 dynallocate() [1/2]

```

template<class B, typename... Args>
B & GiNaC::dynallocate (
    Args &&... args) [inline]
  
```

Constructs a new (class basic or derived) B object on the heap.

This function picks the object's ctor based on the given argument types.

This helps the constructor of ex from basic (or a derived class B) because then the constructor doesn't have to duplicate the object onto the heap. See [ex::construct\\_from\\_basic\(const basic &\)](#) for more information.

References [GiNaC::status\\_flags::dynallocated](#).

Referenced by [abs\\_expand\(\)](#), [GiNaC::numeric::add\\_dyn\(\)](#), [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [base\\_and\\_index\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_star\\_bar\(\)](#), [clifford\\_unit\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_d\(\)](#), [color\\_f\(\)](#), [color\\_ONE\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::diracgammaL::conjugate\(\)](#), [GiNaC::diracgammaR::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#),

[GiNaC::ex::construct\\_from\\_double\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::ex::construct\\_from\\_ulonglong\(\)](#), [delta\\_tensor\(\)](#), [GiNaC::add::derivative\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::ncmul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [diag\\_matrix\(\)](#), [diag\\_matrix\(\)](#), [dirac\\_gamma\(\)](#), [dirac\\_gamma5\(\)](#), [dirac\\_gammaL\(\)](#), [dirac\\_gammaR\(\)](#), [dirac\\_ONE\(\)](#), [dirac\\_slash\(\)](#), [GiNaC::numeric::div\\_dyn\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [epsilon\\_tensor\(\)](#), [epsilon\\_tensor\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::Gt::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::power::evalf\(\)](#), [GiNaC::pseries::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::pseries::evalm\(\)](#), [exadd\(\)](#), [exminus\(\)](#), [exmul\(\)](#), [exp\\_expand\(\)](#), [GiNaC::add::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [gcd\\_pf\\_mul\(\)](#), [get\\_clifford\\_comp\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::ncmul::hold\\_ncmul\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), [index3\(\)](#), [interpolate\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_series\(\)](#), [lorentz\\_eps\(\)](#), [lorentz\\_g\(\)](#), [lst\\_to\\_matrix\(\)](#), [GiNaC::power::map\(\)](#), [GiNaC::relational::map\(\)](#), [metric\\_tensor\(\)](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::symbol::normal\(\)](#), [not\\_symmetric\(\)](#), [pow\(\)](#), [pow\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [reduced\\_matrix\(\)](#), [GiNaC::ncmul::reeval\\_ncmul\(\)](#), [rem\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [replace\\_with\\_symbol\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [spinor\\_metric\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::numeric::sub\\_dyn\(\)](#), [sub\\_matrix\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [symbolic\\_matrix\(\)](#), [symm\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), [symmetric4\(\)](#), [GiNaC::ncmul::thiscontainer\(\)](#), [GiNaC::ncmul::thiscontainer\(\)](#), [GiNaC::add::thisexpairseq\(\)](#), [GiNaC::add::thisexpairseq\(\)](#), [GiNaC::mul::thisexpairseq\(\)](#), [GiNaC::mul::thisexpairseq\(\)](#), and [unit\\_matrix\(\)](#).

#### 7.1.3.15 `dynallocate()` [2/2]

```
template<class B>
B & GiNaC::dynallocate (
    std::initializer_list< ex > il) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function is needed for [GiNaC](#) classes which have public ctors from initializer lists of expressions (which are not a type and not captured by the variadic template version).

References [GiNaC::status\\_flags::dynallocated](#).

#### 7.1.3.16 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [3/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    clifford ,
    indexed ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree )
```

References [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), and [GiNaC::clifford::do\\_print\\_tree\(\)](#).

#### 7.1.3.17 `print_func< print_dflt >()` [1/3]

```
GiNaC::print_func< print_dflt > (
    &diracone::do_print ) &
```

References [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**7.1.3.18 GINAC\_BIND\_UNARCHIVER()** [2/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    clifford )
```

**7.1.3.19 GINAC\_BIND\_UNARCHIVER()** [3/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    cliffordunit )
```

**7.1.3.20 GINAC\_BIND\_UNARCHIVER()** [4/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracone )
```

**7.1.3.21 GINAC\_BIND\_UNARCHIVER()** [5/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma )
```

**7.1.3.22 GINAC\_BIND\_UNARCHIVER()** [6/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma5 )
```

**7.1.3.23 GINAC\_BIND\_UNARCHIVER()** [7/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaL )
```

**7.1.3.24 GINAC\_BIND\_UNARCHIVER()** [8/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaR )
```

**7.1.3.25 is\_dirac\_slash()**

```
bool GiNaC::is_dirac_slash (
    const ex & seq0) [static]
```

References [is\\_a\(\)](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), and [GiNaC::clifford::do\\_print\\_latex\(\)](#).

### 7.1.3.26 base\_and\_index()

```
void GiNaC::base_and_index (
    const ex & c,
    ex & b,
    ex & i) [static]
```

This function decomposes  $\gamma_\mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$ .

References [\\_ex0](#), [\\_ex1](#), [base\\_and\\_index\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), and [GiNaC::varidx::toggle\\_variance\(\)](#).

Referenced by [base\\_and\\_index\(\)](#), [canonicalize\\_clifford\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), and [dirac\\_trace\(\)](#).

### 7.1.3.27 dirac\_ONE()

```
ex GiNaC::dirac_ONE (
    unsigned char rl = 0)
```

Create a Clifford unity object.

#### Parameters

<i>rl</i>	Representation label
-----------	----------------------

#### Returns

newly constructed object

References [dynallocate\(\)](#).

Referenced by [canonicalize\\_clifford\(\)](#), [clifford\\_to\\_lst\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::cliffordunit::contract\\_with\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [dirac\\_trace\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), and [lst\\_to\\_clifford\(\)](#).

### 7.1.3.28 get\_dim\_uint()

```
unsigned GiNaC::get_dim_uint (
    const ex & e) [static]
```

References [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), and [GiNaC::info\\_flags::posint](#).

Referenced by [clifford\\_unit\(\)](#), and [lst\\_to\\_clifford\(\)](#).

### 7.1.3.29 clifford\_unit()

```
ex GiNaC::clifford_unit (
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0)
```

Create a Clifford unit object.

#### Parameters

<i>mu</i>	Index (must be of class <a href="#">varidx</a> or a derived class)
-----------	--

<i>metr</i>	Metric (should be indexed, tensmetric or a derived class, or a matrix)
<i>rl</i>	Representation label

**Returns**

newly constructed Clifford unit object

References [GiNaC::matrix::cols\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [get\\_dim\\_uint\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [is\\_a\(\)](#), [not\\_symmetric\(\)](#), [GiNaC::matrix::rows\(\)](#), and [symmetric2\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#), and [lst\\_to\\_clifford\(\)](#).

**7.1.3.30 dirac\_gamma()**

```
ex GiNaC::dirac_gamma (
    const ex & mu,
    unsigned char rl = 0)
```

Create a Dirac gamma object.

**Parameters**

<i>mu</i>	Index (must be of class varidx or a derived class)
<i>rl</i>	Representation label

**Returns**

newly constructed gamma object

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), and [symmetric2\(\)](#).

**7.1.3.31 dirac\_gamma5()**

```
ex GiNaC::dirac_gamma5 (
    unsigned char rl = 0)
```

Create a Dirac gamma5 object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

References [dynallocate\(\)](#).

Referenced by [dirac\\_trace\(\)](#).



### 7.1.3.32 `dirac_gammaL()`

```
ex GiNaC::dirac_gammaL (
    unsigned char rl = 0)
```

Create a Dirac gammaL object.

#### Parameters

<i>rl</i>	Representation label
-----------	----------------------

#### Returns

newly constructed object

References [dynallocate\(\)](#).

Referenced by [dirac\\_trace\(\)](#).

### 7.1.3.33 `dirac_gammaR()`

```
ex GiNaC::dirac_gammaR (
    unsigned char rl = 0)
```

Create a Dirac gammaR object.

#### Parameters

<i>rl</i>	Representation label
-----------	----------------------

#### Returns

newly constructed object

References [dynallocate\(\)](#).

Referenced by [dirac\\_trace\(\)](#).

### 7.1.3.34 `dirac_slash()`

```
ex GiNaC::dirac_slash (
    const ex & e,
    const ex & dim,
    unsigned char rl = 0)
```

Create a term of the form  $e_\mu * \gamma_\mu$  with a unique index  $\mu$ .

#### Parameters

<i>e</i>	Original expression
----------	---------------------

<i>dim</i>	Dimension of index
<i>rl</i>	Representation label

References [dynallocate\(\)](#), and [symmetric2\(\)](#).

Referenced by [GiNaC::diracgamma::contract\\_with\(\)](#).

#### 7.1.3.35 `get_representation_label()` [1/2]

```
unsigned char GiNaC::get_representation_label (
    const return_type_t & ti) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

References [GiNaC::return\\_type\\_t::rl](#).

Referenced by [canonicalize\\_clifford\(\)](#), [clifford\\_to\\_lst\(\)](#), [color\\_trace\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [dirac\\_trace\(\)](#), [lst\\_to\\_clifford\(\)](#), and [remove\\_dirac\\_ONE\(\)](#).

#### 7.1.3.36 `trace_string()`

```
ex GiNaC::trace_string (
    exvector::const_iterator ix,
    size_t num) [static]
```

Take trace of a string of an even number of Dirac gammas given a vector of indices.

References [lorentz\\_g\(\)](#), and [trace\\_string\(\)](#).

Referenced by [dirac\\_trace\(\)](#), and [trace\\_string\(\)](#).

#### 7.1.3.37 `dirac_trace()` [1/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const std::set< unsigned char > & rls,
    const ex & trONE = 4)
```

Calculate dirac traces over the specified set of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when `gamma5` is involved.

##### Parameters

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

References [\\_ex0](#), [\\_ex1](#), [\\_ex4](#), [base\\_and\\_index\(\)](#), [dirac\\_gamma5\(\)](#), [dirac\\_gammaL\(\)](#), [dirac\\_gammaR\(\)](#), [dirac\\_ONE\(\)](#), [dirac\\_trace\(\)](#), [ex\\_to\(\)](#), [expand\(\)](#), [get\\_representation\\_label\(\)](#), [l\\_is\\_a\(\)](#), [is\\_clifford\\_tinfo\(\)](#), [is\\_exactly\\_a\(\)](#), [lorentz\\_eps\(\)](#), [lorentz\\_g\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [permutation\\_sign\(\)](#), [GiNaC::ex::return\\_type\\_tinfo\(\)](#), [simplify\\_indexed\(\)](#), [GiNaC::ex::subs\(\)](#), and [trace\\_string\(\)](#).

Referenced by [dirac\\_trace\(\)](#), [dirac\\_trace\(\)](#), and [dirac\\_trace\(\)](#).

**7.1.3.38 `dirac_trace()`** [2/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const lst & rll,
    const ex & trONE = 4)
```

Calculate dirac traces over the specified list of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when `gamma5` is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rll</i>	List of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

References [dirac\\_trace\(\)](#), [ex\\_to\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [to\\_int\(\)](#).

**7.1.3.39 `dirac_trace()`** [3/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    unsigned char rl = 0,
    const ex & trONE = 4)
```

Calculate the trace of an expression containing gamma objects with a specified representation label.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when `gamma5` is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

References [dirac\\_trace\(\)](#).

**7.1.3.40 `canonicalize_clifford()`**

```
ex GiNaC::canonicalize_clifford (
    const ex & e)
```

Bring all products of clifford objects in an expression into a canonical order.

This is not necessarily the most simple form but it will allow to check two expressions for equality.

References [\\_ex2](#), [base\\_and\\_index\(\)](#), [canonicalize\\_clifford\(\)](#), [dirac\\_ONE\(\)](#), [ex\\_to\(\)](#), [get\\_representation\\_label\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [is\\_clifford\\_tinfo\(\)](#), [is\\_exactly\\_a\(\)](#), [lhs\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::ex::map\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::return\\_types::noncommutative](#), [rhs\(\)](#), [GiNaC::ex::simplify\\_indexed\(\)](#), [simplify\\_indexed\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

Referenced by [canonicalize\\_clifford\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_to\\_lst\(\)](#), and [remove\\_dirac\\_ONE\(\)](#).

#### 7.1.3.41 clifford\_star\_bar()

```
ex GiNaC::clifford_star_bar (
    const ex & e,
    bool do_bar,
    unsigned options)
```

An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).

References [clifford\\_star\\_bar\(\)](#), [dynallocate\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [clifford\\_bar\(\)](#), [clifford\\_star\(\)](#), and [clifford\\_star\\_bar\(\)](#).

#### 7.1.3.42 clifford\_prime()

```
ex GiNaC::clifford_prime (
    const ex & e)
```

Automorphism of the Clifford algebra, simply changes signs of all clifford units.

References [clifford\\_prime\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [clifford\\_prime\(\)](#), and [clifford\\_to\\_lst\(\)](#).

#### 7.1.3.43 remove\_dirac\_ONE()

```
ex GiNaC::remove_dirac_ONE (
    const ex & e,
    unsigned char rl = 0,
    unsigned options = 0)
```

Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.

For the default value `rl = 0` remove all of them. Aborts if `e` contains any clifford\_unit with `representation_label` to be removed.

##### Parameters

<i>e</i>	Expression to be processed
<i>rl</i>	Value of representation label
<i>options</i>	Defines some internal use

References [canonicalize\\_clifford\(\)](#), [ex\\_to\(\)](#), [expand\\_dummy\\_sum\(\)](#), [get\\_representation\\_label\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [remove\\_dirac\\_ONE\(\)](#).

Referenced by [clifford\\_norm\(\)](#), [clifford\\_to\\_lst\(\)](#), and [remove\\_dirac\\_ONE\(\)](#).

#### 7.1.3.44 clifford\_max\_label()

```
int GiNaC::clifford_max_label (
    const ex & e,
    bool ignore_ONE = false)
```

Returns the maximal representation label of a clifford object if `e` contains at least one, otherwise returns -1.

##### Parameters

<i>e</i>	Expression to be processed
----------	----------------------------

<code>ignore_ONE</code>	defines if clifford_ONE should be ignored in the search
-------------------------	---

References [clifford\\_max\\_label\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [clifford\\_max\\_label\(\)](#), and [GiNaC::add::coeff\(\)](#).

#### 7.1.3.45 clifford\_norm()

```
ex GiNaC::clifford_norm (
    const ex & e)
```

Calculation of the norm in the Clifford algebra.

References [clifford\\_bar\(\)](#), [remove\\_dirac\\_ONE\(\)](#), and [sqrt\(\)](#).

Referenced by [clifford\\_inverse\(\)](#).

#### 7.1.3.46 clifford\_inverse()

```
ex GiNaC::clifford_inverse (
    const ex & e)
```

Calculation of the inverse in the Clifford algebra.

References [clifford\\_bar\(\)](#), [clifford\\_norm\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [pow\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#).

#### 7.1.3.47 lst\_to\_clifford() [1/2]

```
ex GiNaC::lst_to_clifford (
    const ex & v,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0)
```

List or vector conversion into the Clifford vector.

##### Parameters

<i>v</i>	List or vector of coordinates
<i>mu</i>	Index (must be of class varidx or a derived class)
<i>metr</i>	Metric (should be indexed, tensmetric or a derived class, or a matrix)
<i>rl</i>	Representation label

##### Returns

Clifford vector with given components

References [clifford\\_unit\(\)](#), [ex\\_to\(\)](#), and [lst\\_to\\_clifford\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#), and [lst\\_to\\_clifford\(\)](#).

#### 7.1.3.48 lst\_to\_clifford() [2/2]

```
ex GiNaC::lst_to_clifford (  
    const ex & v,  
    const ex & e)
```

List or vector conversion into the Clifford vector.

##### Parameters

$v$	List or vector of coordinates
-----	-------------------------------

<i>e</i>	Clifford unit object
----------	----------------------

**Returns**

Clifford vector with given components

References [cols\(\)](#), [dirac\\_ONE\(\)](#), [ex\\_to\(\)](#), [get\\_dim\\_uint\(\)](#), [get\\_representation\\_label\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [nops\(\)](#), [GiNaC::ex::op\(\)](#), [rows\(\)](#), and [sub\\_matrix\(\)](#).

**7.1.3.49 get\_clifford\_comp()**

```
ex GiNaC::get_clifford_comp (
    const ex & e,
    const ex & c,
    bool root = true) [static]
```

Auxiliary structure to define a function for stripping one Clifford unit from vectors.

Used in [clifford\\_to\\_lst\(\)](#).

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [get\\_clifford\\_comp\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::ex::map\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [clifford\\_to\\_lst\(\)](#), and [get\\_clifford\\_comp\(\)](#).

**7.1.3.50 clifford\_to\_lst()**

```
lst GiNaC::clifford_to_lst (
    const ex & e,
    const ex & c,
    bool algebraic = true)
```

An inverse function to [lst\\_to\\_clifford\(\)](#).

For given Clifford vector extracts its components with respect to given Clifford unit. Obtained components may contain Clifford units with a different metric. Extraction is based on the algebraic formula  $(e * c.i + c.i * e) / \text{pow}(e.i, 2)$  for non-degenerate cases (i.e. neither  $\text{pow}(e.i, 2) = 0$ ).

**Parameters**

<i>e</i>	Clifford expression to be decomposed into components
<i>c</i>	Clifford unit defining the metric for splitting (should have numeric dimension of indices)
<i>algebraic</i>	Use algebraic or symbolic algorithm for extractions

**Returns**

List of components of a Clifford vector

References [GiNaC::container< C >::append\(\)](#), [canonicalize\\_clifford\(\)](#), [clifford\\_prime\(\)](#), [dirac\\_ONE\(\)](#), [ex\\_to\(\)](#), [expand\\_dummy\\_sum\(\)](#), [get\\_clifford\\_comp\(\)](#), [get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [GiNaC::container< C >::remove\\_all\(\)](#), [remove\\_dirac\\_ONE\(\)](#), [simplify\\_indexed\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#).

### 7.1.3.51 clifford\_moebius\_map() [1/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & a,
    const ex & b,
    const ex & c,
    const ex & d,
    const ex & v,
    const ex & G,
    unsigned char rl = 0)
```

Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.

The expression is  $(a * x + b)/(c * x + d)$ , where x is a vector build from list v with metric G. (see Jan Cnops. An introduction to {D}irac operators on manifolds, v.24 of Progress in Mathematical Physics. Birkhauser Boston Inc., Boston, MA, 2002.)

#### Parameters

<i>a</i>	(1,1) entry of the defining matrix
<i>b</i>	(1,2) entry of the defining matrix
<i>c</i>	(2,1) entry of the defining matrix
<i>d</i>	(2,2) entry of the defining matrix
<i>v</i>	Vector to be transformed
<i>G</i>	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
<i>rl</i>	Representation label

#### Returns

List of components of the transformed vector

References [canonicalize\\_clifford\(\)](#), [clifford\\_inverse\(\)](#), [clifford\\_to\\_lst\(\)](#), [clifford\\_unit\(\)](#), [cols\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [G\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [lst\\_to\\_clifford\(\)](#), [rows\(\)](#), and [simplify\\_indexed\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#).

### 7.1.3.52 clifford\_moebius\_map() [2/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & M,
    const ex & v,
    const ex & G,
    unsigned char rl = 0)
```

The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.

#### Parameters

<i>M</i>	the defining matrix
----------	---------------------



$v$	Vector to be transformed
$G$	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
$rl$	Representation label

**Returns**

List of components of the transformed vector

References [clifford\\_moebius\\_map\(\)](#), [cols\(\)](#), [ex\\_to\(\)](#), [G\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::op\(\)](#), and [rows\(\)](#).

**7.1.3.53 GINAC\_DECLARE\_UNARCHIVER()** [2/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    clifford )
```

**7.1.3.54 GINAC\_DECLARE\_UNARCHIVER()** [3/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracone )
```

**7.1.3.55 GINAC\_DECLARE\_UNARCHIVER()** [4/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    cliffordunit )
```

**7.1.3.56 GINAC\_DECLARE\_UNARCHIVER()** [5/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma )
```

**7.1.3.57 GINAC\_DECLARE\_UNARCHIVER()** [6/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma5 )
```

**7.1.3.58 GINAC\_DECLARE\_UNARCHIVER()** [7/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaL )
```

**7.1.3.59 GINAC\_DECLARE\_UNARCHIVER()** [8/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaR )
```

**7.1.3.60 is\_clifford\_tinfo()**

```
bool GiNaC::is_clifford_tinfo (
    const return\_type\_t & ti) [inline]
```

Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).

**Parameters**

<i>ti</i>	tinfo key
-----------	-----------

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [canonicalize\\_clifford\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), and [dirac\\_trace\(\)](#).

**7.1.3.61 clifford\_bar()**

```
ex GiNaC::clifford_bar (
    const ex & e) [inline]
```

Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.

References [clifford\\_star\\_bar\(\)](#).

Referenced by [clifford\\_inverse\(\)](#), and [clifford\\_norm\(\)](#).

**7.1.3.62 clifford\_star()**

```
ex GiNaC::clifford_star (
    const ex & e) [inline]
```

Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.

References [clifford\\_star\\_bar\(\)](#).

**7.1.3.63 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [4/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    su3one ,
    tensor ,
    print_func< print\_dflt > &::do_print. print_func< print\_latex > &::do_print_↵
    latex )
```

References [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**7.1.3.64 print\_func< print\_dflt >() [2/3]**

```
GiNaC::print_func< print\_dflt > (
    &su3t::do\_print ) &
```

References [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**7.1.3.65 GINAC\_BIND\_UNARCHIVER()** [9/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    color )
```

**7.1.3.66 GINAC\_BIND\_UNARCHIVER()** [10/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3one )
```

**7.1.3.67 GINAC\_BIND\_UNARCHIVER()** [11/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3t )
```

**7.1.3.68 GINAC\_BIND\_UNARCHIVER()** [12/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3f )
```

**7.1.3.69 GINAC\_BIND\_UNARCHIVER()** [13/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3d )
```

References [is\\_a\(\)](#).

**7.1.3.70 permute\_free\_index\_to\_front()**

```
ex GiNaC::permute_free_index_to_front (
    const exvector & iv3,
    const exvector & iv2,
    int & sig) [static]
```

Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.

**Parameters**

<i>iv3</i>	Vector of 3 indices
<i>iv2</i>	Vector of 2 indices, must be a subset of iv3
<i>sig</i>	Returns sign introduced by index permutation

**Returns**

the free index (the one that is in iv3 but not in iv2)

References [GINAC\\_ASSERT](#), and [TEST\\_PERMUTATION](#).

Referenced by [GiNaC::su3d::contract\\_with\(\)](#), and [GiNaC::su3f::contract\\_with\(\)](#).

**7.1.3.71 color\_ONE()**

```
ex GiNaC::color_ONE (
    unsigned char rl = 0)
```

Create the su(3) unity element.

This is an indexed object, although it has no indices.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed unity element

References [dynallocate\(\)](#).

Referenced by [GiNaC::su3t::contract\\_with\(\)](#).

**7.1.3.72 color\_T()**

```
ex GiNaC::color_T (
    const ex & a,
    unsigned char rl = 0)
```

Create an su(3) generator.

**Parameters**

<i>a</i>	Index
<i>rl</i>	Representation label

**Returns**

newly constructed unity generator

References [dynallocate\(\)](#), [ex\\_to\(\)](#), and [is\\_a\(\)](#).

Referenced by [color\\_trace\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), and [GiNaC::su3f::contract\\_with\(\)](#).

**7.1.3.73 color\_f()**

```
ex GiNaC::color_f (
    const ex & a,
    const ex & b,
    const ex & c)
```

Create an su(3) antisymmetric structure constant.

**Parameters**

<i>a</i>	First index
----------	-------------

<i>b</i>	Second index
<i>c</i>	Third index

#### Returns

newly constructed structure constant

References [antisymmetric3\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), and [is\\_a\(\)](#).

Referenced by [color\\_h\(\)](#).

#### 7.1.3.74 `color_d()`

```
ex GiNaC::color_d (  
    const ex & a,  
    const ex & b,  
    const ex & c)
```

Create an su(3) symmetric structure constant.

#### Parameters

<i>a</i>	First index
<i>b</i>	Second index
<i>c</i>	Third index

#### Returns

newly constructed structure constant

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), and [symmetric3\(\)](#).

Referenced by [color\\_h\(\)](#).

#### 7.1.3.75 `color_h()`

```
ex GiNaC::color_h (  
    const ex & a,  
    const ex & b,  
    const ex & c)
```

This returns the linear combination  $d.a.b.c + l.f.a.b.c$ .

References [color\\_d\(\)](#), [color\\_f\(\)](#), and [l](#).

Referenced by [color\\_trace\(\)](#).

**7.1.3.76 is\_color\_tinfo()**

```
bool GiNaC::is_color_tinfo (
    const return_type_t & ti) [static]
```

Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [color\\_trace\(\)](#).

**7.1.3.77 get\_representation\_label()** [2/2]

```
unsigned char GiNaC::get_representation_label (
    const return_type_t & ti) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

References [GiNaC::return\\_type\\_t::rl](#).

**7.1.3.78 color\_trace()** [1/3]

```
ex GiNaC::color_trace (
    const ex & e,
    const std::set< unsigned char > & rls)
```

Calculate color traces over the specified set of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels

References [\\_ex0](#), [\\_ex1](#), [\\_ex3](#), [color\\_h\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [delta\\_tensor\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_representation\\_label\(\)](#), [is\\_a\(\)](#), [is\\_color\\_tinfo\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

Referenced by [color\\_trace\(\)](#), [color\\_trace\(\)](#), [color\\_trace\(\)](#), and [GiNaC::su3t::contract\\_with\(\)](#).

**7.1.3.79 color\_trace()** [2/3]

```
ex GiNaC::color_trace (
    const ex & e,
    const lst & rll)
```

Calculate color traces over the specified list of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
----------	---------------------------------

<i>rl</i>	List of representation labels
-----------	-------------------------------

References [color\\_trace\(\)](#), [ex\\_to\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [to\\_int\(\)](#).

#### 7.1.3.80 color\_trace() [3/3]

```
ex GiNaC::color_trace (
    const ex & e,
    unsigned char rl = 0)
```

Calculate the trace of an expression containing color objects with a specified representation label.

##### Parameters

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label

References [color\\_trace\(\)](#).

#### 7.1.3.81 GINAC\_DECLARE\_UNARCHIVER() [9/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    color )
```

#### 7.1.3.82 GINAC\_DECLARE\_UNARCHIVER() [10/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3one )
```

#### 7.1.3.83 GINAC\_DECLARE\_UNARCHIVER() [11/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3t )
```

#### 7.1.3.84 GINAC\_DECLARE\_UNARCHIVER() [12/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3f )
```

#### 7.1.3.85 GINAC\_DECLARE\_UNARCHIVER() [13/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3d )
```

**7.1.3.86 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [5/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    constant ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr >
&::do_print_python_repr ) const
```

References [GiNaC::constant::do\\_print\(\)](#), [GiNaC::constant::do\\_print\\_latex\(\)](#), [GiNaC::constant::do\\_print\\_python\\_repr\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**7.1.3.87 GINAC\_BIND\_UNARCHIVER()** [14/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    constant )
```

**7.1.3.88 GINAC\_DECLARE\_UNARCHIVER()** [14/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    constant )
```

**7.1.3.89 crc32()**

```
unsigned GiNaC::crc32 (
    const char * c,
    const unsigned len,
    const unsigned crcinit) [static]
```

References [crctab](#).

**7.1.3.90 are\_ex\_trivially\_equal()**

```
bool GiNaC::are_ex_trivially_equal (
    const ex & e1,
    const ex & e2) [inline]
```

Compare two objects of class quickly without doing a deep tree traversal.

**Returns**

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

Referenced by [GiNaC::add::conjugate\(\)](#), [GiNaC::container< std::vector >::conjugate\(\)](#), [GiNaC::expair::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::Gt::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::make\\_flat\\_inserter::make\\_flat\\_inserter\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::power::map\(\)](#), [GiNaC::relational::map\(\)](#), [GiNaC::const\\_iterator::operator==\(\)](#), [GiNaC::internal::\\_iter\\_rep::operator==\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), and [GiNaC::expairseq::subschildren\(\)](#).



**7.1.3.91 operator<<() [3/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exvector & e)
```

References [get\\_print\\_context\(\)](#).

**7.1.3.92 operator<<() [4/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exset & e)
```

References [get\\_print\\_context\(\)](#).

**7.1.3.93 operator<<() [5/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exmap & e)
```

References [get\\_print\\_context\(\)](#).

**7.1.3.94 nops() [1/2]**

```
size_t GiNaC::nops (
    const ex & thisex) [inline]
```

References [GiNaC::ex::nops\(\)](#).

Referenced by [GiNaC::container< C >::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [lst\\_to\\_clifford\(\)](#), and [GiNaC::container< C >::op\(\)](#).

**7.1.3.95 expand() [1/2]**

```
ex GiNaC::expand (
    const ex & thisex,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::expand\(\)](#).

Referenced by [dirac\\_trace\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [log\\_expand\(\)](#), [prem\(\)](#), [quo\(\)](#), [rem\(\)](#), and [sprem\(\)](#).

**7.1.3.96 conjugate()**

```
ex GiNaC::conjugate (
    const ex & thisex) [inline]
```

References [GiNaC::ex::conjugate\(\)](#).

Referenced by [conjugate\\_expl\\_derivative\(\)](#).

### 7.1.3.97 real\_part()

```
ex GiNaC::real_part (
    const ex & thisex) [inline]
```

References [GiNaC::ex::real\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [log\\_imag\\_part\(\)](#), [real\\_part\\_expl\\_derivative\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 7.1.3.98 imag\_part()

```
ex GiNaC::imag_part (
    const ex & thisex) [inline]
```

References [GiNaC::ex::imag\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [imag\\_part\\_expl\\_derivative\(\)](#), [log\\_imag\\_part\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 7.1.3.99 has()

```
bool GiNaC::has (
    const ex & thisex,
    const ex & pattern,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::has\(\)](#).

### 7.1.3.100 find()

```
bool GiNaC::find (
    const ex & thisex,
    const ex & pattern,
    exset & found) [inline]
```

References [GiNaC::ex::find\(\)](#).

### 7.1.3.101 is\_polynomial()

```
bool GiNaC::is_polynomial (
    const ex & thisex,
    const ex & vars) [inline]
```

References [GiNaC::ex::is\\_polynomial\(\)](#).

**7.1.3.102 degree()**

```
int GiNaC::degree (
    const ex & thisex,
    const ex & s) [inline]
```

References [GiNaC::ex::degree\(\)](#).

Referenced by [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**7.1.3.103 ldegree()**

```
int GiNaC::ldegree (
    const ex & thisex,
    const ex & s) [inline]
```

References [GiNaC::ex::ldegree\(\)](#).

**7.1.3.104 coeff()**

```
ex GiNaC::coeff (
    const ex & thisex,
    const ex & s,
    int n = 1) [inline]
```

References [GiNaC::ex::coeff\(\)](#).

Referenced by [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [log\\_series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**7.1.3.105 numer() [1/2]**

```
ex GiNaC::numer (
    const ex & thisex) [inline]
```

References [GiNaC::ex::numer\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [print\\_real\\_csrc\(\)](#), [GiNaC::power::real\\_part\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**7.1.3.106 denom() [1/2]**

```
ex GiNaC::denom (
    const ex & thisex) [inline]
```

References [GiNaC::ex::denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcmcoeff\(\)](#), [print\\_real\\_csrc\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 7.1.3.107 numer\_denom()

```
ex GiNaC::numer_denom (  
    const ex & thisex) [inline]
```

References [GiNaC::ex::numer\\_denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 7.1.3.108 normal()

```
ex GiNaC::normal (  
    const ex & thisex) [inline]
```

References [GiNaC::ex::normal\(\)](#).

Referenced by [fsolve\(\)](#), [GiNaC::normal\\_map\\_function::operator\(\)\(\)](#), and [replace\\_with\\_symbol\(\)](#).

#### 7.1.3.109 to\_rational()

```
ex GiNaC::to_rational (  
    const ex & thisex,  
    exmap & repl) [inline]
```

References [GiNaC::ex::to\\_rational\(\)](#).

#### 7.1.3.110 to\_polynomial()

```
ex GiNaC::to_polynomial (  
    const ex & thisex,  
    exmap & repl) [inline]
```

References [GiNaC::ex::to\\_polynomial\(\)](#).

#### 7.1.3.111 collect()

```
ex GiNaC::collect (  
    const ex & thisex,  
    const ex & s,  
    bool distributed = false) [inline]
```

References [GiNaC::ex::collect\(\)](#).

#### 7.1.3.112 eval()

```
ex GiNaC::eval (  
    const ex & thisex) [inline]
```

References [GiNaC::ex::eval\(\)](#).

**7.1.3.113 evalf()** [1/2]

```
ex GiNaC::evalf (
    const ex & thisex) [inline]
```

References [GiNaC::ex::evalf\(\)](#).

Referenced by [beta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [GiNaC::evalf\\_map\\_function::operator\(\)](#)(), and [zeta2\\_evalf\(\)](#).

**7.1.3.114 evalm()**

```
ex GiNaC::evalm (
    const ex & thisex) [inline]
```

References [GiNaC::ex::evalm\(\)](#).

Referenced by [GiNaC::evalm\\_map\\_function::operator\(\)](#)().

**7.1.3.115 eval\_integ()**

```
ex GiNaC::eval_integ (
    const ex & thisex) [inline]
```

References [GiNaC::ex::eval\\_integ\(\)](#).

Referenced by [GiNaC::eval\\_integ\\_map\\_function::operator\(\)](#)().

**7.1.3.116 diff()**

```
ex GiNaC::diff (
    const ex & thisex,
    const symbol & s,
    unsigned nth = 1) [inline]
```

References [GiNaC::ex::diff\(\)](#).

Referenced by [GiNaC::derivative\\_map\\_function::operator\(\)](#)().

**7.1.3.117 series()**

```
ex GiNaC::series (
    const ex & thisex,
    const ex & r,
    int order,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::series\(\)](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [Li2\\_series\(\)](#), and [log\\_series\(\)](#).

**7.1.3.118 match()**

```
bool GiNaC::match (
    const ex & thisex,
    const ex & pattern,
    exmap & repl_lst) [inline]
```

References [GiNaC::ex::match\(\)](#).

**7.1.3.119 simplify\_indexed() [1/3]**

```
ex GiNaC::simplify_indexed (
    const ex & thisex,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::simplify\\_indexed\(\)](#).

Referenced by [canonicalize\\_clifford\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_to\\_lst\(\)](#), [dirac\\_trace\(\)](#), [GiNaC::ex::simplify\\_indexed\(\)](#), [GiNaC::ex::simplify\\_indexed\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.120 simplify\_indexed() [2/3]**

```
ex GiNaC::simplify_indexed (
    const ex & thisex,
    const scalar_products & sp,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::simplify\\_indexed\(\)](#).

**7.1.3.121 symmetrize() [1/4]**

```
ex GiNaC::symmetrize (
    const ex & thisex) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

Referenced by [idx\\_symmetrization\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**7.1.3.122 symmetrize() [2/4]**

```
ex GiNaC::symmetrize (
    const ex & thisex,
    const lst & l) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

**7.1.3.123 antisymmetrize()** [1/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

Referenced by [antisymmetrize\(\)](#), and [GiNaC::ex::antisymmetrize\(\)](#).

**7.1.3.124 antisymmetrize()** [2/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex,
    const lst & l) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

**7.1.3.125 symmetrize\_cyclic()** [1/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::ex::symmetrize\\_cyclic\(\)](#), and [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**7.1.3.126 symmetrize\_cyclic()** [2/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex,
    const lst & l) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**7.1.3.127 op()**

```
ex GiNaC::op (
    const ex & thisex,
    size_t i) [inline]
```

References [GiNaC::ex::op\(\)](#).

Referenced by [get\\_clifford\\_comp\(\)](#), [Li2\\_series\(\)](#), and [rename\\_dummy\\_indices\(\)](#).

**7.1.3.128 lhs()**

```
ex GiNaC::lhs (
    const ex & thisex) [inline]
```

References [GiNaC::ex::lhs\(\)](#).

Referenced by [canonicalize\\_clifford\(\)](#), [GiNaC::ptr< T >::operator!=](#), [GiNaC::ptr< T >::operator!=](#), [GiNaC::const\\_iterator::operator-](#), [GiNaC::ptr< T >::operator==](#), and [GiNaC::ptr< T >::operator==](#).

**7.1.3.129 rhs()**

```
ex GiNaC::rhs (
    const ex & thisex) [inline]
```

References [GiNaC::ex::rhs\(\)](#).

Referenced by [canonicalize\\_clifford\(\)](#), [lsolve\(\)](#), [GiNaC::ptr< T >::operator!=](#), [GiNaC::ptr< T >::operator!==\(](#), [GiNaC::ptr< T >::operator!=](#), [GiNaC::const\\_iterator::operator-](#), [GiNaC::ptr< T >::operator<<](#), [GiNaC::ptr< T >::operator==](#), [GiNaC::ptr< T >::operator==\(](#), [GiNaC::ptr< T >::operator==](#), [GiNaC::matrix::solve\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**7.1.3.130 is\_zero()** [1/2]

```
bool GiNaC::is_zero (
    const ex & thisex) [inline]
```

References [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [cosh\\_eval\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::pathintegral\\_term\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::Gt::regularise\(\)](#), [sinh\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

**7.1.3.131 swap()** [1/2]

```
void GiNaC::swap (
    ex & e1,
    ex & e2) [inline]
```

References [GiNaC::ex::swap\(\)](#).

Referenced by [permutation\\_sign\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

**7.1.3.132 subs()** [1/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const exmap & m,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::subs\(\)](#).

Referenced by [Li2\\_series\(\)](#).

**7.1.3.133 subs()** [2/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const lst & ls,
    const lst & lr,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::subs\(\)](#).



**7.1.3.134 subs()** [3/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const ex & e,
    unsigned options = 0) [inline]
```

References [GiNaC::ex::subs\(\)](#).

**7.1.3.135 is\_a()** [2/3]

```
template<class T>
bool GiNaC::is_a (
    const ex & obj) [inline]
```

Check if ex is a handle to a T, including base classes.

**7.1.3.136 is\_exactly\_a()** [2/2]

```
template<class T>
bool GiNaC::is_exactly_a (
    const ex & obj) [inline]
```

Check if ex is a handle to a T, not including base classes.

**7.1.3.137 ex\_to()**

```
template<class T>
attribute_pure const T & GiNaC::ex_to (
    const ex & e) [inline]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of e first. Also, you shouldn't cache the returned reference because [GiNaC's](#) garbage collector may destroy the referenced object any time it's used in another expression.

**Parameters**

<i>e</i>	expression
----------	------------

**Returns**

reference to object of class T

See also

[is\\_exactly\\_a<class T>\(\)](#)

Referenced by [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_power\(\)](#), [acos\\_eval\(\)](#), [acos\\_evalf\(\)](#), [acosh\\_eval\(\)](#), [acosh\\_evalf\(\)](#), [adaptivesimpson\(\)](#), [GiNaC::symmetry::add\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [GiNaC::Gt::apply\\_function\\_recursive\(\)](#), [asin\\_eval\(\)](#), [asin\\_evalf\(\)](#), [asinh\\_conjugate\(\)](#), [asinh\\_eval\(\)](#), [asinh\\_evalf\(\)](#), [atan2\\_eval\(\)](#), [atan2\\_evalf\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_eval\(\)](#), [atan\\_evalf\(\)](#), [atan\\_series\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_evalf\(\)](#), [atanh\\_series\(\)](#), [base\\_and\\_index\(\)](#), [beta\\_eval\(\)](#), [beta\\_evalf\(\)](#), [beta\\_series\(\)](#), [binomial\\_eval\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::symmetry::canonicalize\(\)](#), [canonicalize\\_clifford\(\)](#), [clifford\\_max\\_label\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_moebius\\_map\(\)](#), [clifford\\_to\\_lst\(\)](#), [clifford\\_unit\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_d\(\)](#), [color\\_f\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [color\\_trace\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [conjugate\\_evalf\(\)](#), [conjugate\\_expl\\_derivative\(\)](#), [GiNaC::expairseq::construct\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_expairseq\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::cliffordunit::contract\\_with\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [csgn\\_power\(\)](#), [csgn\\_series\(\)](#), [GiNaC::basic::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::pseries::degree\(\)](#), [dirac\\_gamma\(\)](#), [dirac\\_trace\(\)](#), [dirac\\_trace\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticK\\_evalf\(\)](#), [epsilon\\_tensor\(\)](#), [epsilon\\_tensor\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [exp\\_eval\(\)](#), [exp\\_evalf\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [expand\\_dummy\\_sum\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [factorial\\_eval\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [frac\\_cancel\(\)](#), [fsolve\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [get\\_clifford\\_comp\(\)](#), [get\\_dim\\_uint\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::Gt::Gt\(\)](#), [GiNaC::Gt::Gt\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [H\\_evalf\(\)](#), [H\\_print\\_latex\(\)](#), [GiNaC::numeric::has\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::symmetry::has\\_cyclic\(\)](#), [GiNaC::symmetry::has\\_nonsymmetric\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [imag\\_part\\_evalf\(\)](#), [imag\\_part\\_expl\\_derivative\(\)](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), [index3\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\(\)](#), [is\\_dummy\\_pair\(\)](#), [GiNaC::basic::is\\_polynomial\(\)](#), [is\\_the\\_function\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [kronecker\\_symbol\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [GiNaC::basic::ldegree\(\)](#), [GiNaC::mul::ldegree\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), [GiNaC::pseries::ldegree\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_evalf\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_evalf\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [Li\\_print\\_latex\(\)](#), [log\\_eval\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [lorentz\\_eps\(\)](#), [lsolve\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [lst\\_to\\_clifford\(\)](#), [lst\\_to\\_clifford\(\)](#), [lst\\_to\\_matrix\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::Gt::match\(\)](#), [GiNaC::wildcard::match\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [not\\_symmetric\(\)](#), [GiNaC::relational::operator\\_safe\\_bool\(\)](#), [GiNaC::idx\\_is\\_equal\\_ignore\\_dim::operator\(\)](#), [GiNaC::sy\\_is\\_less::operator\(\)](#), [GiNaC::sy\\_swap::operator\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [Order\\_eval\(\)](#), [Order\\_series\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::indexed::printindices\(\)](#), [psi1\\_eval\(\)](#), [psi1\\_evalf\(\)](#), [psi1\\_series\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_evalf\(\)](#), [psi2\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::Eisenstein\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::symmetry::read\\_archive\(\)](#), [GiNaC::power::real\\_part\(\)](#), [real\\_part\\_evalf\(\)](#), [real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [remove\\_dirac\\_ONE\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices\(\)](#), [S\\_eval\(\)](#), [S\\_evalf\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::Ebar\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::ELi\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::series\\_coeff\\_impl\(\)](#), [series\\_to\\_poly\(\)](#), [GiNaC::Gt::setArgs\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [simplify\\_indexed\(\)](#)

[GiNaC::indexed::simplify\\_indexed\\_product](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [spinor\\_metric\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [sqrtfree\(\)](#), [sqrtfree\\_parfrac\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [step\\_series\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), [symmetric4\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tgamma\\_eval\(\)](#), [tgamma\\_evalf\(\)](#), [tgamma\\_series\(\)](#), [GiNaC::indexed::thiscontainer\(\)](#), [GiNaC::indexed::thiscontainer\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), [GiNaC::Gt::to\\_numeric\(\)](#), [tryfactsubs\(\)](#), [GiNaC::indexed::validate\(\)](#), [zeta1\\_eval\(\)](#), [zeta1\\_evalf\(\)](#), [zeta1\\_print\\_latex\(\)](#), [zeta2\\_eval\(\)](#), [zeta2\\_evalf\(\)](#), and [zeta2\\_print\\_latex\(\)](#).

### 7.1.3.138 compile\_ex() [1/3]

```

void GiNaC::compile_ex (
    const ex & expr,
    const symbol & sym,
    FUNCP\_1P & fp,
    const std::string filename = "")
  
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type [FUNCP\\_1P](#).

#### Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

### 7.1.3.139 compile\_ex() [2/3]

```

void GiNaC::compile_ex (
    const ex & expr,
    const symbol & sym1,
    const symbol & sym2,
    FUNCP\_2P & fp,
    const std::string filename = "")
  
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type [FUNCP\\_2P](#).

#### Parameters

<i>expr</i>	Expression to be compiled
<i>sym1</i>	Symbol from the expression to become the first function parameter
<i>sym2</i>	Symbol from the expression to become the second function parameter
<i>fp</i>	Returned function pointer

<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted
-----------------	---

#### 7.1.3.140 `compile_ex()` [3/3]

```
void GiNaC::compile_ex (
    const lst & exprs,
    const lst & syms,
    FUNCP_CUBA & fp,
    const std::string filename = "")
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type [FUNCP\\_CUBA](#).

##### Parameters

<i>exprs</i>	List of expression to be compiled
<i>syms</i>	Symbols from the expression to become the function parameters
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

#### 7.1.3.141 `link_ex()` [1/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_1P & fp)
```

Opens an existing so-file and returns a function pointer of type [FUNCP\\_1P](#) to the contained function.

The so-file has to be generated by `compile_ex` in advance.

##### Parameters

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

#### 7.1.3.142 `link_ex()` [2/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_2P & fp)
```

Opens an existing so-file and returns a function pointer of type [FUNCP\\_2P](#) to the contained function.

The so-file has to be generated by `compile_ex` in advance.

##### Parameters

<i>filename</i>	Name of the so-file to open and link
-----------------	--------------------------------------

<i>fp</i>	Returned function pointer
-----------	---------------------------

#### 7.1.3.143 `link_ex()` [3/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNC_P_CUBA & fp)
```

Opens an existing so-file and returns a function pointer of type [FUNC\\_P\\_CUBA](#) to the contained function.

The so-file has to be generated by `compile_ex` in advance.

##### Parameters

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

#### 7.1.3.144 `unlink_ex()`

```
void GiNaC::unlink_ex (
    const std::string filename)
```

Closes all linked .so files that have the supplied filename.

##### Parameters

<i>filename</i>	Name of the so-file to close
-----------------	------------------------------

#### 7.1.3.145 `swap()` [2/2]

```
void GiNaC::swap (
    expair & e1,
    expair & e2) [inline]
```

References [GiNaC::expair::swap\(\)](#).

#### 7.1.3.146 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [6/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    expairseq ,
    basic ,
    print_func< print_context > &::do_print, print_func< print_tree > &::do_print↵
    _tree )
```

References [GiNaC::expairseq::do\\_print\(\)](#), and [GiNaC::expairseq::do\\_print\\_tree\(\)](#).

### 7.1.3.147 conjugateepvector()

```
epvector * GiNaC::conjugateepvector (
    const epvector & )
```

Complex conjugate every element of an epvector.

Returns zero if this does not change anything.

References [GiNaC::expair::conjugate\(\)](#), and [GiNaC::expair::is\\_equal\(\)](#).

Referenced by [GiNaC::expairseq::conjugate\(\)](#), and [GiNaC::pseries::conjugate\(\)](#).

### 7.1.3.148 factor()

```
ex GiNaC::factor (
    const ex & poly,
    unsigned options)
```

Interface function to the outside world.

Factorizes univariate and multivariate polynomials.

It uses [factor1\(\)](#) on each of the explicitly present factors of poly.

The default option is [factor\\_options::polynomial](#), which means that [factor\(\)](#) will only factorize an expression if it is a proper polynomial (i.e. the flag [info\\_flags::polynomial](#) is set). Given the option [factor\\_options::all](#), [factor\(\)](#) will factorize all subexpressions, e.g. polynomials containing functions or polynomials inside function arguments.

#### Parameters

in	<i>poly</i>	expression to factorize
in	<i>options</i>	see <a href="#">GiNaC::factor_options</a>

#### Returns

factorized expression

References [pow\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [collect\\_common\\_factors\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::mul::series\(\)](#), [sqfree\\_parfrac\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

### 7.1.3.149 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [7/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    fail ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
    _tree )
```

References [GiNaC::fail::do\\_print\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), and [GINAC\\_BIND\\_UNARCHIVER](#).

**7.1.3.150 GINAC\_DECLARE\_UNARCHIVER()** [15/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    fail )
```

**7.1.3.151 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [8/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    fderivative ,
    function ,
    print_func< print_context > &::do_print, print_func< print_latex > &::do_↵
print_latex, print_func< print_csrc > &::do_print_csrc, print_func< print_tree > &::do_↵
print_tree )
```

References [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::fderivative::do\\_print\\_tree\(\)](#).

**7.1.3.152 GINAC\_BIND\_UNARCHIVER()** [15/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    fderivative )
```

**7.1.3.153 GINAC\_DECLARE\_UNARCHIVER()** [16/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    fderivative )
```

**7.1.3.154 GINAC\_BIND\_UNARCHIVER()** [16/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    function )
```

**7.1.3.155 GINAC\_DECLARE\_UNARCHIVER()** [17/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    function )
```

**7.1.3.156 is\_the\_function()**

```
template<typename T>
bool GiNaC::is_the_function (
    const ex & x) [inline]
```

References [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [is\\_the\\_function< G\\_SERIAL >\(\)](#), [is\\_the\\_function< iterated\\_integral\\_SERIAL >\(\)](#), [is\\_the\\_function< psi\\_SERIAL >\(\)](#), [is\\_the\\_function< zeta\\_SERIAL >\(\)](#), and [GiNaC::Gt\\_detail::TransformExpressionWithCache< GiNaC::G2\\_SERIAL >::print\\_cache\(\)](#).

**7.1.3.157 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [9/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Gt ,
    basic ,
    print_func< print_dflt > &::do_print.  print_func< print_python > &::do_print.
    print_func< print_latex > &::do_print_latex )
```

References [GiNaC::Gt::do\\_print\(\)](#), and [GiNaC::Gt::do\\_print\\_latex\(\)](#).

**7.1.3.158 GINAC\_BIND\_UNARCHIVER()** [17/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Gt )
```

**7.1.3.159 GINAC\_DECLARE\_UNARCHIVER()** [18/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Gt )
```

**7.1.3.160 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [10/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    idx ,
    basic ,
    print_func< print_context > &::do_print.  print_func< print_latex > &::do_↵
    print_latex.  print_func< print_csrc > &::do_print_csrc.  print_func< print_tree > &::do_↵
    print_tree )
```

References [GiNaC::idx::do\\_print\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_latex\(\)](#), and [GiNaC::idx::do\\_print\\_tree\(\)](#).

**7.1.3.161 print\_func< print\_context >()**

```
GiNaC::print_func< print_context > (
    &varidx::do_print ) &
```

**7.1.3.162 GINAC\_BIND\_UNARCHIVER()** [18/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    idx )
```

**7.1.3.163 GINAC\_BIND\_UNARCHIVER()** [19/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    varidx )
```



**7.1.3.164 GINAC\_BIND\_UNARCHIVER()** [20/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinidx )
```

**7.1.3.165 is\_dummy\_pair()** [1/2]

```
bool GiNaC::is_dummy_pair (
    const idx & i1,
    const idx & i2)
```

Check whether two indices form a dummy pair.

References [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#).

Referenced by [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [GiNaC::indexed::has\\_dummy\\_index\\_for\(\)](#), [is\\_dummy\\_pair\(\)](#), [GiNaC::is\\_summation\\_idx::operator\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), and [GiNaC::indexed::reposition\\_dummy\\_indices](#).

**7.1.3.166 is\_dummy\_pair()** [2/2]

```
bool GiNaC::is_dummy_pair (
    const ex & e1,
    const ex & e2)
```

Check whether two expressions form a dummy index pair.

References [ex\\_to\(\)](#), [is\\_a\(\)](#), and [is\\_dummy\\_pair\(\)](#).

**7.1.3.167 find\_free\_and\_dummy()** [1/2]

```
void GiNaC::find_free_and_dummy (
    exvector::const_iterator it,
    exvector::const_iterator itend,
    exvector & out_free,
    exvector & out_dummy)
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>it</i>	Pointer to start of index vector
<i>itend</i>	Pointer to end of index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References [ex\\_to\(\)](#), [is\\_dummy\\_pair\(\)](#), and [shaker\\_sort\(\)](#).

Referenced by [GiNaC::su3d::contract\\_with\(\)](#), [count\\_dummy\\_indices\(\)](#), [count\\_free\\_indices\(\)](#), [find\\_dummy\\_indices\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::indexed::get\\_dummy\\_indices\(\)](#), [GiNaC::indexed::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.168 minimal\_dim()**

```
ex GiNaC::minimal_dim (
    const ex & dim1,
    const ex & dim2)
```

Return the minimum of two index dimensions.

If this is undecidable, throw an exception. Numeric dimensions are always considered "smaller" than symbolic dimensions.

References [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.169 GINAC\_DECLARE\_UNARCHIVER()** [19/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    idx )
```

**7.1.3.170 GINAC\_DECLARE\_UNARCHIVER()** [20/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    varidx )
```

**7.1.3.171 GINAC\_DECLARE\_UNARCHIVER()** [21/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinidx )
```

**7.1.3.172 find\_free\_and\_dummy()** [2/2]

```
void GiNaC::find_free_and_dummy (
    const exvector & v,
    exvector & out_free,
    exvector & out_dummy) [inline]
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>v</i>	Index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References [find\\_free\\_and\\_dummy\(\)](#).

### 7.1.3.173 find\_dummy\_indices()

```
void GiNaC::find_dummy_indices (
    const exvector & v,
    exvector & out_dummy) [inline]
```

Given a vector of indices, find the dummy indices.

#### Parameters

<i>v</i>	Index vector
----------	--------------

<i>out_dummy</i>	Vector of dummy indices (returned, sorted)
------------------	--

References [find\\_free\\_and\\_dummy\(\)](#).

Referenced by [GiNaC::indexed::get\\_dummy\\_indices\(\)](#).

### 7.1.3.174 count\_dummy\_indices()

```
size_t GiNaC::count_dummy_indices (
    const exvector & v) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

### 7.1.3.175 count\_free\_indices()

```
size_t GiNaC::count_free_indices (
    const exvector & v) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

### 7.1.3.176 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [11/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    indexed ,
    exprseq ,
    print_func< print\_context > &::do_print. print_func< print\_latex > &::do_←
print_latex. print_func< print\_tree > &::do_print_tree )
```

References [GiNaC::indexed::do\\_print\(\)](#), [GiNaC::indexed::do\\_print\\_latex\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), and [not\\_symmetric\(\)](#).

### 7.1.3.177 GINAC\_BIND\_UNARCHIVER() [21/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    indexed )
```

### 7.1.3.178 indices\_consistent()

```
bool GiNaC::indices_consistent (
    const exvector & v1,
    const exvector & v2) [static]
```

Check whether two sorted index vectors are consistent (i.e.

equal).

Referenced by [GiNaC::add::get\\_free\\_indices\(\)](#), and [GiNaC::indexed::simplify\\_indexed](#).

**7.1.3.179 number\_of\_type()**

```
template<class T>
size_t GiNaC::number_of_type (
    const exvector & v)
```

References [is\\_exactly\\_a\(\)](#).

Referenced by [rename\\_dummy\\_indices\(\)](#).

**7.1.3.180 rename\_dummy\_indices()**

```
template<class T>
ex GiNaC::rename_dummy_indices (
    const ex & e,
    exvector & global_dummy_indices,
    exvector & local_dummy_indices) [static]
```

Rename dummy indices in an expression.

**Parameters**

<i>e</i>	Expression to work on
<i>local_dummy_indices</i>	The set of dummy indices that appear in the expression "e"
<i>global_dummy_indices</i>	The set of dummy indices that have appeared before and which we would like to use in "e", too. This gets updated by the function

References [GINAC\\_ASSERT](#), [is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [number\\_of\\_type\(\)](#), [op\(\)](#), [shaker\\_sort\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.181 find\_variant\_indices()**

```
void GiNaC::find_variant_indices (
    const exvector & v,
    exvector & variant_indices) [static]
```

Given a set of indices, extract those of class `varidx`.

References [is\\_exactly\\_a\(\)](#).

Referenced by [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.182 reposition\_dummy\_indices()**

```
bool GiNaC::reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices)
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

**Parameters**

<i>e</i>	Object to work on
----------	-------------------

<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)
<i>moved_indices</i>	The set of indices that have been repositioned (will be changed by this function)

**Returns**

true if 'e' was changed

Referenced by [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.183 product\_to\_exvector()**

```
void GiNaC::product_to_exvector (
    const ex & e,
    exvector & v,
    bool & non_commutative) [static]
```

References [\\_ex2](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [get\\_all\\_dummy\\_indices\(\)](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.184 idx\_symmetrization()**

```
template<class T>
ex GiNaC::idx_symmetrization (
    const ex & r,
    const exvector & local_dummy_indices)
```

References [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [symmetrize\(\)](#).

Referenced by [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.185 simplify\_indexed() [3/3]**

```
ex GiNaC::simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar\_products & sp)
```

Simplify indexed expression, return list of free indices.

**7.1.3.186 simplify\_indexed\_product()**

```
ex GiNaC::simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar\_products & sp)
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

Referenced by [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::indexed::simplify\\_indexed\\_product](#).

**7.1.3.187 hasindex()**

```
bool GiNaC::hasindex (
    const ex & x,
    const ex & sym)
```

References [hasindex\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [hasindex\(\)](#), and [GiNaC::indexed::simplify\\_indexed](#).

**7.1.3.188 get\_all\_dummy\_indices\_safely()**

```
exvector GiNaC::get_all_dummy_indices_safely (
    const ex & e)
```

More reliable version of the form.

The former assumes that e is an expanded expression.

References [ex\\_to\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::mul::expand\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

**7.1.3.189 get\_all\_dummy\_indices()**

```
exvector GiNaC::get_all_dummy_indices (
    const ex & e)
```

Returns all dummy indices from the exvector.

Returns all dummy indices from the expression.

References [ex\\_to\(\)](#), [is\\_a\(\)](#), and [product\\_to\\_exvector\(\)](#).

Referenced by [expand\\_dummy\\_sum\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), and [get\\_clifford\\_comp\(\)](#).

**7.1.3.190 rename\_dummy\_indices\_uniquely()** [1/4]

```
lst GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb)
```

Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

**7.1.3.191 rename\_dummy\_indices\_uniquely()** [2/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb,
    const ex & b)
```

Same as above, where va and vb contain the indices of a and b and are sorted.

References [ex\\_to\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.192 rename\_dummy\_indices\_uniquely()** [3/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const ex & a,
    const ex & b)
```

Returns b with all dummy indices, which are common with a, renamed.

References [ex\\_to\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.193 rename\_dummy\_indices\_uniquely()** [4/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    exvector & va,
    const ex & b,
    bool modify_va)
```

Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.

References [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [ex\\_to\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.194 expand\_dummy\_sum()**

```
ex GiNaC::expand_dummy_sum (
    const ex & e,
    bool subs_idx = false)
```

This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.

Optionally all indices with a variance will be substituted by indices with the corresponding numeric values without variance.

**Parameters**

<i>e</i>	the given expression
----------	----------------------



<code>subs_idx</code>	indicates if variance of dummy indices should be neglected
-----------------------	--

References [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\\_dummy\\_sum\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [idx](#), [is\\_a\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [clifford\\_to\\_lst\(\)](#), [expand\\_dummy\\_sum\(\)](#), and [remove\\_dirac\\_ONE\(\)](#).

#### 7.1.3.195 GINAC\_DECLARE\_UNARCHIVER() [22/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    indexed )
```

#### 7.1.3.196 conjugate\_evalf()

```
ex GiNaC::conjugate_evalf (
    const ex & arg) [static]
```

References [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.197 conjugate\_eval()

```
ex GiNaC::conjugate_eval (
    const ex & arg) [static]
```

References [GiNaC::ex::conjugate\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.198 conjugate\_print\_latex()

```
void GiNaC::conjugate_print_latex (
    const ex & arg,
    const print_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.199 conjugate\_conjugate()

```
ex GiNaC::conjugate_conjugate (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.200 conjugate\_expl\_derivative()**

```
ex GiNaC::conjugate_expl_derivative (
    const ex & arg,
    const symbol & s) [static]
```

References [conjugate\(\)](#), [GiNaC::ex::diff\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.201 conjugate\_real\_part()**

```
ex GiNaC::conjugate_real_part (
    const ex & arg) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.202 conjugate\_imag\_part()**

```
ex GiNaC::conjugate_imag_part (
    const ex & arg) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.203 func\_arg\_info()**

```
bool GiNaC::func_arg_info (
    const ex & arg,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [conjugate\\_info\(\)](#).

**7.1.3.204 conjugate\_info()**

```
bool GiNaC::conjugate_info (
    const ex & arg,
    unsigned inf) [static]
```

References [func\\_arg\\_info\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.205 REGISTER\_FUNCTION()** [1/36]

```
GiNaC::REGISTER_FUNCTION (
    conjugate_function ,
    eval_func(conjugate_eval). evalf_func(conjugate_evalf). expl_derivative_↵
func(conjugate_expl_derivative). info_func(conjugate_info). print_func< print_latex >(conjugate_print_latex
conjugate_func(conjugate_conjugate). real_part_func(conjugate_real_part). imag_part_func(conjugate_imag_part
set_name("conjugate","conjugate") )
```

References [conjugate\\_conjugate\(\)](#), [conjugate\\_eval\(\)](#), [conjugate\\_evalf\(\)](#), [conjugate\\_expl\\_derivative\(\)](#), [conjugate\\_imag\\_part\(\)](#), [conjugate\\_info\(\)](#), [conjugate\\_print\\_latex\(\)](#), and [conjugate\\_real\\_part\(\)](#).

**7.1.3.206 real\_part\_evalf()**

```
ex GiNaC::real_part_evalf (
    const ex & arg) [static]
```

References [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.207 real\_part\_eval()**

```
ex GiNaC::real_part_eval (
    const ex & arg) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.208 real\_part\_print\_latex()**

```
void GiNaC::real_part_print_latex (
    const ex & arg,
    const print_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.209 real\_part\_conjugate()**

```
ex GiNaC::real_part_conjugate (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.210 real\_part\_real\_part()**

```
ex GiNaC::real_part_real_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.211 real\_part\_imag\_part()**

```
ex GiNaC::real_part_imag_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.212 real\_part\_expl\_derivative()**

```
ex GiNaC::real_part_expl_derivative (
    const ex & arg,
    const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), [GiNaC::info\\_flags::real](#), and [real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.213 REGISTER\_FUNCTION() [2/36]**

```
GiNaC::REGISTER_FUNCTION (
    real_part_function ,
    eval_func(real_part_eval). evalf_func(real_part_evalf). expl_derivative_←
func(real_part_expl_derivative). print_func< print_latex >(real_part_print_latex). conjugate_←
_func(real_part_conjugate). real_part_func(real_part_real_part). imag_part_func(real_part_imag_part).
set_name("real_part","real_part") )
```

References [real\\_part\\_conjugate\(\)](#), [real\\_part\\_eval\(\)](#), [real\\_part\\_evalf\(\)](#), [real\\_part\\_expl\\_derivative\(\)](#), [real\\_part\\_imag\\_part\(\)](#), [real\\_part\\_print\\_latex\(\)](#), and [real\\_part\\_real\\_part\(\)](#).

**7.1.3.214 imag\_part\_evalf()**

```
ex GiNaC::imag_part_evalf (
    const ex & arg) [static]
```

References [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.215 `imag_part_eval()`**

```
ex GiNaC::imag_part_eval (
    const ex & arg) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.216 `imag_part_print_latex()`**

```
void GiNaC::imag_part_print_latex (
    const ex & arg,
    const print_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.217 `imag_part_conjugate()`**

```
ex GiNaC::imag_part_conjugate (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.218 `imag_part_real_part()`**

```
ex GiNaC::imag_part_real_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.219 `imag_part_imag_part()`**

```
ex GiNaC::imag_part_imag_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.220 `imag_part_expl_derivative()`**

```
ex GiNaC::imag_part_expl_derivative (
    const ex & arg,
    const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\\_part\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.221 REGISTER\_FUNCTION() [3/36]**

```
GiNaC::REGISTER_FUNCTION (
    imag_part_function ,
    eval_func(imag_part_eval). evalf_func(imag_part_evalf). expl_derivative_↔
func(imag_part_expl_derivative). print_func< print_latex >(imag_part_print_latex). conjugate_↔
_func(imag_part_conjugate). real_part_func(imag_part_real_part). imag_part_func(imag_part_imag_part).
set_name("imag_part","imag_part") )
```

References [imag\\_part\\_conjugate\(\)](#), [imag\\_part\\_eval\(\)](#), [imag\\_part\\_evalf\(\)](#), [imag\\_part\\_expl\\_derivative\(\)](#), [imag\\_part\\_imag\\_part\(\)](#), [imag\\_part\\_print\\_latex\(\)](#), and [imag\\_part\\_real\\_part\(\)](#).

**7.1.3.222 abs\_evalf()**

```
ex GiNaC::abs_evalf (
    const ex & arg) [static]
```

References [abs\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.223 abs\_eval()**

```
ex GiNaC::abs_eval (
    const ex & arg) [static]
```

References [abs\(\)](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.224 abs\_expand()**

```
ex GiNaC::abs_expand (
    const ex & arg,
    unsigned options) [static]
```

References [abs\(\)](#), [GiNaC::ex::begin\(\)](#), [dynallocate\(\)](#), [GiNaC::ex::end\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.225 abs\_expl\_derivative()**

```
ex GiNaC::abs_expl_derivative (
    const ex & arg,
    const symbol & s) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), and [GiNaC::ex::diff\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.226 `abs_print_latex()`

```
void GiNaC::abs_print_latex (  
    const ex & arg,  
    const print\_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.227 `abs_print_csrc_float()`

```
void GiNaC::abs_print_csrc_float (  
    const ex & arg,  
    const print\_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.228 `abs_conjugate()`

```
ex GiNaC::abs_conjugate (  
    const ex & arg) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.229 `abs_real_part()`

```
ex GiNaC::abs_real_part (  
    const ex & arg) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.230 `abs_imag_part()`

```
ex GiNaC::abs_imag_part (  
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.231 abs\_power()**

```
ex GiNaC::abs_power (
    const ex & arg,
    const ex & exp) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::info\\_flags::even](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_even\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.232 abs\_info()**

```
bool GiNaC::abs_info (
    const ex & arg,
    unsigned inf)
```

References [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.233 REGISTER\_FUNCTION() [4/36]**

```
GiNaC::REGISTER_FUNCTION (
    abs ,
    eval_func(abs_eval). evalf_func(abs_evalf). expand_func(abs_expand). expl_↵
    derivative_func(abs_expl_derivative). info_func(abs_info). print_func< print_latex >(abs_print_latex).
    print_func< print_csrc_float >(abs_print_csrc_float). print_func< print_csrc_double >(abs_print_csrc_float)
    conjugate_func(abs_conjugate). real_part_func(abs_real_part). imag_part_func(abs_imag_part).
    power_func(abs_power) )
```

References [abs\(\)](#), [abs\\_conjugate\(\)](#), [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_expand\(\)](#), [abs\\_expl\\_derivative\(\)](#), [abs\\_imag\\_part\(\)](#), [abs\\_info\(\)](#), [abs\\_power\(\)](#), [abs\\_print\\_csrc\\_float\(\)](#), [abs\\_print\\_latex\(\)](#), and [abs\\_real\\_part\(\)](#).

**7.1.3.234 step\_evalf()**

```
ex GiNaC::step_evalf (
    const ex & arg) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.235 step\_eval()**

```
ex GiNaC::step_eval (
    const ex & arg) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::numeric::imag\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::numeric::real\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.236 step\_series()**

```
ex GiNaC::step_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [step\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.237 step\_conjugate()**

```
ex GiNaC::step_conjugate (
    const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.238 step\_real\_part()**

```
ex GiNaC::step_real_part (
    const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.239 step\_imag\_part()**

```
ex GiNaC::step_imag_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.240 REGISTER\_FUNCTION() [5/36]**

```
GiNaC::REGISTER_FUNCTION (
    step ,
    eval_func(step_eval). evalf_func(step_evalf). series_func(step_series). conjugate←
_func(step_conjugate). real_part_func(step_real_part). imag_part_func(step_imag_part) )
```

References [step\(\)](#), [step\\_conjugate\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [step\\_imag\\_part\(\)](#), [step\\_real\\_part\(\)](#), and [step\\_series\(\)](#).

**7.1.3.241 csgn\_evalf()**

```
ex GiNaC::csgn_evalf (
    const ex & arg) [static]
```

References [csgn\(\)](#), [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.242 csgn\_eval()**

```
ex GiNaC::csgn_eval (
    const ex & arg) [static]
```

References [csgn\(\)](#), [ex\\_to\(\)](#), [I](#), [GiNaC::numeric::imag\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::numeric::real\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.243 csgn\_series()**

```
ex GiNaC::csgn_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.244 csgn\_conjugate()**

```
ex GiNaC::csgn_conjugate (
    const ex & arg) [static]
```

References [csgn\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.245 csgn\_real\_part()**

```
ex GiNaC::csgn_real_part (
    const ex & arg) [static]
```

References [csgn\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.246 csgn\_imag\_part()**

```
ex GiNaC::csgn_imag_part (
    const ex & arg) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.247 csgn\_power()**

```
ex GiNaC::csgn_power (
    const ex & arg,
    const ex & exp) [static]
```

References [\\_ex2](#), [csgn\(\)](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_a\(\)](#), [is\\_odd\(\)](#), and [GiNaC::info\\_flags::positive](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.248 REGISTER\_FUNCTION() [6/36]**

```
GiNaC::REGISTER_FUNCTION (
    csgn ,
    eval_func(csgn_eval). evalf_func(csgn_evalf). series_func(csgn_series). conjugate↔
_func(csgn_conjugate). real_part_func(csgn_real_part). imag_part_func(csgn_imag_part). power↔
_func(csgn_power) )
```

References [csgn\(\)](#), [csgn\\_conjugate\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [csgn\\_imag\\_part\(\)](#), [csgn\\_power\(\)](#), [csgn\\_real\\_part\(\)](#), and [csgn\\_series\(\)](#).

**7.1.3.249 eta\_evalf()**

```
ex GiNaC::eta_evalf (
    const ex & x,
    const ex & y) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [evalf\(\)](#), [ex\\_to\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [GiNaC::info\\_flags::positive](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.250 eta\_eval()**

```
ex GiNaC::eta_eval (
    const ex & x,
    const ex & y) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [ex\\_to\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [GiNaC::info\\_flags::positive](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.251 eta\_series()**

```
ex GiNaC::eta_series (
    const ex & x,
    const ex & y,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.252 eta\_conjugate()**

```
ex GiNaC::eta_conjugate (
    const ex & x,
    const ex & y) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.253 eta\_real\_part()**

```
ex GiNaC::eta_real_part (
    const ex & x,
    const ex & y) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.254 eta\_imag\_part()**

```
ex GiNaC::eta_imag_part (
    const ex & x,
    const ex & y) [static]
```

References [I](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.255 REGISTER\_FUNCTION() [7/36]**

```
GiNaC::REGISTER_FUNCTION (
    eta ,
    eval_func(eta_eval). evalf_func(eta_evalf). series_func(eta_series). latex↵
_name("\\eta"). set_symmetry(sy_symm(0, 1)). conjugate_func(eta_conjugate). real_part_↵
func(eta_real_part). imag_part_func(eta_imag_part) )
```

References [eta\\_conjugate\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [eta\\_imag\\_part\(\)](#), [eta\\_real\\_part\(\)](#), and [eta\\_series\(\)](#).

**7.1.3.256 Li2\_evalf()**

```
ex GiNaC::Li2_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [Li2\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.257 Li2\_eval()**

```
ex GiNaC::Li2_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex12](#), [\\_ex1\\_2](#), [\\_ex2](#), [\\_ex6](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_48](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.258 Li2\_deriv()**

```
ex GiNaC::Li2_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), and [log\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.259 Li2\_series() [1/2]**

```
ex GiNaC::Li2_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_num2\\_p](#), [ex\\_to\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::relational::lhs\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [Pi](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), [subs\(\)](#), [GiNaC::series\\_options::suppress\\_branchcut](#), and [zeta\(\)](#).

Referenced by [Li2\\_projection\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.260 Li2\_conjugate()**

```
ex GiNaC::Li2_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), and [GiNaC::info\\_flags::negative](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.261 REGISTER\_FUNCTION() [8/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li2 ,
    eval_func(Li2_eval). evalf_func(Li2_evalf). derivative_func(Li2_deriv). series←
_func(Li2_series). conjugate_func(Li2_conjugate). latex_name("\\mathrm{Li}_2") )
```

References [Li2\(\)](#), [Li2\\_conjugate\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

**7.1.3.262 Li3\_eval()**

```
ex GiNaC::Li3_eval (
    const ex & x) [static]
```

References [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.263 REGISTER\_FUNCTION() [9/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li3 ,
    eval_func(Li3_eval). latex_name("\\mathrm{Li}_3") )
```

References [Li3\\_eval\(\)](#).

**7.1.3.264 zetaderiv\_eval()**

```
ex GiNaC::zetaderiv_eval (
    const ex & n,
    const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::numeric](#), and [zeta\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.265 zetaderiv\_deriv()**

```
ex GiNaC::zetaderiv_deriv (
    const ex & n,
    const ex & x,
    unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.266 REGISTER\_FUNCTION()** [10/36]

```
GiNaC::REGISTER_FUNCTION (
    zetaderiv ,
    eval_func(zetaderiv_eval). derivative_func(zetaderiv_deriv). latex_name("\\zeta^\\prime")
)
```

References [zetaderiv\\_deriv\(\)](#), and [zetaderiv\\_eval\(\)](#).

**7.1.3.267 factorial\_evalf()**

```
ex GiNaC::factorial_evalf (
    const ex & x) [static]
```

References [factorial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.268 factorial\_eval()**

```
ex GiNaC::factorial_eval (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.269 factorial\_print\_dflt\_latex()**

```
void GiNaC::factorial_print_dflt_latex (
    const ex & x,
    const print_context & c) [static]
```

References [is\\_exactly\\_a\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.270 factorial\_conjugate()**

```
ex GiNaC::factorial_conjugate (
    const ex & x) [static]
```

References [factorial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.271 factorial\_real\_part()**

```
ex GiNaC::factorial_real_part (
    const ex & x) [static]
```

References [factorial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.272 factorial\_imag\_part()**

```
ex GiNaC::factorial_imag_part (
    const ex & x) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.273 REGISTER\_FUNCTION() [11/36]**

```
GiNaC::REGISTER_FUNCTION (
    factorial ,
    eval_func(factorial_eval). evalf_func(factorial_evalf). print_func< print_dflt
>(factorial_print_dflt_latex). print_func< print_latex >(factorial_print_dflt_latex). conjugate←
_func(factorial_conjugate). real_part_func(factorial_real_part). imag_part_func(factorial_imag_part)
)
```

References [factorial\(\)](#), [factorial\\_conjugate\(\)](#), [factorial\\_eval\(\)](#), [factorial\\_evalf\(\)](#), [factorial\\_imag\\_part\(\)](#), [factorial\\_print\\_dflt\\_latex\(\)](#), and [factorial\\_real\\_part\(\)](#).

**7.1.3.274 binomial\_evalf()**

```
ex GiNaC::binomial_evalf (
    const ex & x,
    const ex & y) [static]
```

References [binomial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.275 binomial\_sym()**

```
ex GiNaC::binomial_sym (
    const ex & x,
    const numeric & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [binomial\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [binomial\\_evalf\(\)](#).



**7.1.3.276 binomial\_eval()**

```
ex GiNaC::binomial_eval (
    const ex & x,
    const ex & y) [static]
```

References [binomial\(\)](#), [binomial\\_sym\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [is\\_integer\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.277 binomial\_conjugate()**

```
ex GiNaC::binomial_conjugate (
    const ex & x,
    const ex & y) [static]
```

References [binomial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.278 binomial\_real\_part()**

```
ex GiNaC::binomial_real_part (
    const ex & x,
    const ex & y) [static]
```

References [binomial\(\)](#), and [GiNaC::basic::hold\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.279 binomial\_imag\_part()**

```
ex GiNaC::binomial_imag_part (
    const ex & x,
    const ex & y) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.280 REGISTER\_FUNCTION() [12/36]**

```
GiNaC::REGISTER_FUNCTION (
    binomial ,
    eval_func(binomial_eval) . evalf_func(binomial_evalf) . conjugate_func(binomial_conjugate) .
    real_part_func(binomial_real_part) . imag_part_func(binomial_imag_part) )
```

References [binomial\(\)](#), [binomial\\_conjugate\(\)](#), [binomial\\_eval\(\)](#), [binomial\\_evalf\(\)](#), [binomial\\_imag\\_part\(\)](#), and [binomial\\_real\\_part\(\)](#).

**7.1.3.281 Order\_eval()**

```
ex GiNaC::Order_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [ex\\_to\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.282 Order\_series()**

```
ex GiNaC::Order_series (
    const ex & x,
    const relational & r,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::relational::lhs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.283 Order\_conjugate()**

```
ex GiNaC::Order_conjugate (
    const ex & x) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.284 Order\_real\_part()**

```
ex GiNaC::Order_real_part (
    const ex & x) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.285 Order\_imag\_part()**

```
ex GiNaC::Order_imag_part (
    const ex & x) [static]
```

References [GiNaC::ex::info\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.286 Order\_power()**

```
ex GiNaC::Order_power (
    const ex & x,
    const ex & e) [static]
```

References [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::info\\_flags::posint](#), and [pow\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.287 Order\_expl\_derivative()**

```
ex GiNaC::Order_expl_derivative (
    const ex & arg,
    const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.288 REGISTER\_FUNCTION()** [13/36]

```
GiNaC::REGISTER_FUNCTION (
    Order ,
    eval_func(Order_eval). series_func(Order_series). latex_name("\\mathcal{O}").
    expl_derivative_func(Order_expl_derivative). conjugate_func(Order_conjugate). real_part_↔
    func(Order_real_part). imag_part_func(Order_imag_part). power_func(Order_power) )
```

References [Order\\_conjugate\(\)](#), [Order\\_eval\(\)](#), [Order\\_expl\\_derivative\(\)](#), [Order\\_imag\\_part\(\)](#), [Order\\_power\(\)](#), [Order\\_real\\_part\(\)](#), and [Order\\_series\(\)](#).

**7.1.3.289 Isolve()**

```
ex GiNaC::lsolve (
    const ex & eqns,
    const ex & symbols,
    unsigned options = solve_algo::automatic)
```

Factorial function.

Binomial function. Order term function (for truncated power series).

References [GiNaC::container< C >::append\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::matrix::cols\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::info\\_flags::exprseq](#), [GINAC\\_ASSERT](#), [GiNaC::symbolset::has\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::info\\_flags::list](#), [Isolve\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::relation\\_equal](#), [rhs\(\)](#), [GiNaC::matrix::rows\(\)](#), [GiNaC::matrix::solve\(\)](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [Isolve\(\)](#).

**7.1.3.290 fsolve()**

```
const numeric GiNaC::fsolve (
    const ex & f,
    const symbol & x,
    const numeric & x1,
    const numeric & x2)
```

Find a real root of real-valued function f(x) numerically within a given interval.

The function must change sign across interval. Uses Newton- Raphson method combined with bisection in order to guarantee convergence.

**Parameters**

<i>f</i>	Function f(x)
----------	---------------

$x$	Symbol $f(x)$
$x1$	lower interval limit
$x2$	upper interval limit

### Exceptions

<code>runtime_error</code>	(if interval is invalid).
----------------------------	---------------------------

References [GiNaC::ex::diff\(\)](#), [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [is\\_real\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::lhs\(\)](#), [normal\(\)](#), [GiNaC::ex::rhs\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 7.1.3.291 `zeta()` [1/3]

```
template<typename T1>
function GiNaC::zeta (
    const T1 & p1) [inline]
```

References [GiNaC::zeta1\\_SERIAL::serial](#).

Referenced by [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [S\\_eval\(\)](#), [zeta1\\_eval\(\)](#), [zeta1\\_evalf\(\)](#), [zeta2\\_eval\(\)](#), [zeta2\\_evalf\(\)](#), and [zetaderiv\\_eval\(\)](#).

#### 7.1.3.292 `zeta()` [2/3]

```
template<typename T1, typename T2>
function GiNaC::zeta (
    const T1 & p1,
    const T2 & p2) [inline]
```

References [GiNaC::zeta2\\_SERIAL::serial](#).

#### 7.1.3.293 `is_the_function<zeta_SERIAL>()`

```
template<>
bool GiNaC::is_the_function<zeta_SERIAL> (
    const ex & x) [inline]
```

References [is\\_the\\_function\(\)](#).

#### 7.1.3.294 `G()` [1/2]

```
template<typename T1, typename T2>
function GiNaC::G (
    const T1 & x,
    const T2 & y) [inline]
```

References [GiNaC::G2\\_SERIAL::serial](#).

Referenced by [clifford\\_moebius\\_map\(\)](#), [clifford\\_moebius\\_map\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), and [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#).

**7.1.3.295 G()** [2/2]

```
template<typename T1, typename T2, typename T3>
function GiNaC::G (
    const T1 & x,
    const T2 & s,
    const T3 & y) [inline]
```

References [GiNaC::G3\\_SERIAL::serial](#).

**7.1.3.296 is\_the\_function< G\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< G_SERIAL > (
    const ex & x) [inline]
```

References [is\\_the\\_function\(\)](#).

**7.1.3.297 psi()** [1/4]

```
template<typename T1>
function GiNaC::psi (
    const T1 & p1) [inline]
```

References [GiNaC::psi1\\_SERIAL::serial](#).

Referenced by [beta\\_deriv\(\)](#), [lgamma\\_deriv\(\)](#), [psi1\\_deriv\(\)](#), [psi1\\_eval\(\)](#), [psi1\\_evalf\(\)](#), [psi1\\_series\(\)](#), [psi2\\_deriv\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_evalf\(\)](#), [psi2\\_series\(\)](#), [sr\\_gcd\(\)](#), and [tgamma\\_deriv\(\)](#).

**7.1.3.298 psi()** [2/4]

```
template<typename T1, typename T2>
function GiNaC::psi (
    const T1 & p1,
    const T2 & p2) [inline]
```

References [GiNaC::psi2\\_SERIAL::serial](#).

**7.1.3.299 is\_the\_function< psi\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< psi_SERIAL > (
    const ex & x) [inline]
```

References [is\\_the\\_function\(\)](#).

**7.1.3.300 iterated\_integral()** [1/2]

```
template<typename T1, typename T2>
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda) [inline]
```

References [GiNaC::iterated\\_integral2\\_SERIAL::serial](#).

Referenced by [iterated\\_integral2\\_eval\(\)](#), [iterated\\_integral3\\_eval\(\)](#), and [iterated\\_integral\\_evalf\\_impl\(\)](#).

**7.1.3.301 iterated\_integral()** [2/2]

```
template<typename T1, typename T2, typename T3>
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda,
    const T3 & N_trunc) [inline]
```

References [GiNaC::iterated\\_integral3\\_SERIAL::serial](#).

**7.1.3.302 is\_the\_function< iterated\_integral\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< iterated_integral_SERIAL > (
    const ex & x) [inline]
```

References [is\\_the\\_function\(\)](#).

**7.1.3.303 is\_order\_function()**

```
bool GiNaC::is_order_function (
    const ex & e) [inline]
```

Check whether a function is the Order (O(n)) function.

References [is\\_ex\\_the\\_function](#).

Referenced by [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), and [GiNaC::integral::series\(\)](#).

**7.1.3.304 convert\_H\_to\_Li()**

```
ex GiNaC::convert_H_to_Li (
    const ex & parameterlst,
    const ex & arg)
```

Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.

References [is\\_a\(\)](#).

**7.1.3.305 EllipticK\_evalf()**

```
ex GiNaC::EllipticK_evalf (
    const ex & k) [static]
```

References [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.306 EllipticK\_eval()**

```
ex GiNaC::EllipticK_eval (
    const ex & k) [static]
```

References [\\_ex0](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.307 EllipticK\_deriv()**

```
ex GiNaC::EllipticK_deriv (
    const ex & k,
    unsigned deriv_param) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.308 EllipticK\_series()**

```
ex GiNaC::EllipticK_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::subs\\_options::no\\_pattern](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.309 EllipticK\_print\_latex()**

```
void GiNaC::EllipticK_print_latex (
    const ex & k,
    const print_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.310 REGISTER\_FUNCTION() [14/36]**

```
GiNaC::REGISTER_FUNCTION (
    EllipticK ,
    evalf_func(EllipticK_evalf). eval_func(EllipticK_eval). derivative_func(EllipticK_deriv).
    series_func(EllipticK_series). print_func< print_latex >(EllipticK_print_latex). do_not_↔
    evalf_params() )
```

References [EllipticK\\_deriv\(\)](#), [EllipticK\\_eval\(\)](#), [EllipticK\\_evalf\(\)](#), [EllipticK\\_print\\_latex\(\)](#), and [EllipticK\\_series\(\)](#).

**7.1.3.311 EllipticE\_evalf()**

```
ex GiNaC::EllipticE_evalf (
    const ex & k) [static]
```

References [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.312 EllipticE\_eval()**

```
ex GiNaC::EllipticE_eval (
    const ex & k) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.313 EllipticE\_deriv()**

```
ex GiNaC::EllipticE_deriv (
    const ex & k,
    unsigned deriv_param) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.314 EllipticE\_series()**

```
ex GiNaC::EllipticE_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::subs\\_options::no\\_pattern](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.315 EllipticE\_print\_latex()**

```
void GiNaC::EllipticE_print_latex (
    const ex & k,
    const print\_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.316 REGISTER\_FUNCTION() [15/36]**

```
GiNaC::REGISTER_FUNCTION (
    EllipticE ,
    evalf_func(EllipticE\_evalf). eval_func(EllipticE\_eval). derivative_func(EllipticE\_deriv).
    series_func(EllipticE\_series). print_func< print\_latex >(EllipticE\_print\_latex). do_not_↵
    evalf_params() )
```

References [EllipticE\\_deriv\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticE\\_print\\_latex\(\)](#), and [EllipticE\\_series\(\)](#).

**7.1.3.317 iterated\_integral\_evalf\_impl()**

```
ex GiNaC::iterated_integral_evalf_impl (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc) [static]
```

References [coeff\(\)](#), [Digits](#), [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [iterated\\_integral\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::info\\_flags::numeric](#).

Referenced by [iterated\\_integral2\\_evalf\(\)](#), and [iterated\\_integral3\\_evalf\(\)](#).

**7.1.3.318 iterated\_integral2\_evalf()**

```
ex GiNaC::iterated_integral2_evalf (
    const ex & kernel_lst,
    const ex & lambda) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**7.1.3.319 iterated\_integral3\_evalf()**

```
ex GiNaC::iterated_integral3_evalf (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**7.1.3.320 iterated\_integral2\_eval()**

```
ex GiNaC::iterated_integral2_eval (
    const ex & kernel_lst,
    const ex & lambda) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**7.1.3.321 iterated\_integral3\_eval()**

```
ex GiNaC::iterated_integral3_eval (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**7.1.3.322 lgamma\_evalf()**

```
ex GiNaC::lgamma_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [is\\_exactly\\_a\(\)](#), and [lgamma\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.323 lgamma\_eval()**

```
ex GiNaC::lgamma_eval (
    const ex & x) [static]
```

Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.

Handles integer arguments as a special case.

**Exceptions**

<a href="#">GiNaC::pole_error("lgamma_eval()")</a>	logarithmic pole",0)
--	----------------------

References [\\_ex\\_1](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_rational\(\)](#), [lgamma\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::info\\_flags::posint](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.324 lgamma\_deriv()**

```
ex GiNaC::lgamma_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), and [psi\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.325 lgamma\_series()**

```
ex GiNaC::lgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [lgamma\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::positive](#), [GiNaC::basic::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.326 lgamma\_conjugate()**

```
ex GiNaC::lgamma_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lgamma\(\)](#), and [GiNaC::info\\_flags::positive](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.327 REGISTER\_FUNCTION() [16/36]**

```
GiNaC::REGISTER_FUNCTION (
    lgamma ,
    eval_func(lgamma_eval). evalf_func(lgamma_evalf). derivative_func(lgamma_deriv).
    series_func(lgamma_series). conjugate_func(lgamma_conjugate). latex_name("\\log \\Gamma") )
```

References [lgamma\(\)](#), [lgamma\\_conjugate\(\)](#), [lgamma\\_deriv\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_evalf\(\)](#), and [lgamma\\_series\(\)](#).

**7.1.3.328 tgamma\_evalf()**

```
ex GiNaC::tgamma_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [is\\_exactly\\_a\(\)](#), and [tgamma\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

### 7.1.3.329 `tgamma_eval()`

```
ex GiNaC::tgamma_eval (
    const ex & x) [static]
```

Evaluation of `tgamma(x)`, the true Gamma function.

Knows about integer arguments, half-integer arguments and that's it. Somebody ought to provide some good numerical evaluation some day...

#### Exceptions

<code>pole_error("tgamma_eval() simple pole",0)</code>	
--	--

References `_num1_2_p`, `_num1_p`, `_num2_p`, `_num_2_p`, `abs()`, `GiNaC::numeric::div()`, `doublefactorial()`, `ex_to()`, `factorial()`, `GiNaC::ex::info()`, `GiNaC::numeric::is_even()`, `GiNaC::numeric::is_integer()`, `GiNaC::numeric::is_positive()`, `is_rational()`, `GiNaC::numeric::mul()`, `GiNaC::info_flags::numeric`, `Pi`, `pow()`, `sqrt()`, `GiNaC::numeric::sub()`, and `tgamma()`.

Referenced by `REGISTER_FUNCTION()`.

### 7.1.3.330 `tgamma_deriv()`

```
ex GiNaC::tgamma_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References `GINAC_ASSERT`, `psi()`, and `tgamma()`.

Referenced by `REGISTER_FUNCTION()`.

### 7.1.3.331 `tgamma_series()`

```
ex GiNaC::tgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References `_ex1`, `ex_to()`, `GiNaC::ex::info()`, `GiNaC::info_flags::integer`, `GiNaC::subs_options::no_pattern`, `GiNaC::info_flags::positive`, `GiNaC::ex::series()`, `GiNaC::ex::subs()`, and `tgamma()`.

Referenced by `REGISTER_FUNCTION()`.

### 7.1.3.332 `tgamma_conjugate()`

```
ex GiNaC::tgamma_conjugate (
    const ex & x) [static]
```

References `GiNaC::ex::conjugate()`, and `tgamma()`.

Referenced by `REGISTER_FUNCTION()`.

**7.1.3.333 REGISTER\_FUNCTION() [17/36]**

```
GiNaC::REGISTER_FUNCTION (
    tgamma ,
    eval_func(tgamma_eval). evalf_func(tgamma_evalf). derivative_func(tgamma_deriv).
    series_func(tgamma_series). conjugate_func(tgamma_conjugate). latex_name("\\Gamma") )
```

References [tgamma\(\)](#), [tgamma\\_conjugate\(\)](#), [tgamma\\_deriv\(\)](#), [tgamma\\_eval\(\)](#), [tgamma\\_evalf\(\)](#), and [tgamma\\_series\(\)](#).

**7.1.3.334 beta\_evalf()**

```
ex GiNaC::beta_evalf (
    const ex & x,
    const ex & y) [static]
```

References [ex\\_to\(\)](#), [exp\(\)](#), [is\\_exactly\\_a\(\)](#), and [lgamma\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.335 beta\_eval()**

```
ex GiNaC::beta_eval (
    const ex & x,
    const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_num\\_1\\_p](#), [evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [is\\_positive\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), and [tgamma\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.336 beta\_deriv()**

```
ex GiNaC::beta_deriv (
    const ex & x,
    const ex & y,
    unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), and [psi\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.337 beta\_series()**

```
ex GiNaC::beta_series (
    const ex & arg1,
    const ex & arg2,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_a\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::positive](#), [GiNaC::ex::subs\(\)](#), and [tgamma\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.338 REGISTER\_FUNCTION() [18/36]**

```
GiNaC::REGISTER_FUNCTION (
    beta ,
    eval_func(beta_eval). evalf_func(beta_evalf). derivative_func(beta_deriv).
    series_func(beta_series). latex_name("\\mathrm{B}"). set_symmetry(sy_symm(0, 1)) )
```

References [beta\\_deriv\(\)](#), [beta\\_eval\(\)](#), [beta\\_evalf\(\)](#), and [beta\\_series\(\)](#).

**7.1.3.339 psi1\_evalf()**

```
ex GiNaC::psi1_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [psi\(\)](#).

**7.1.3.340 psi1\_eval()**

```
ex GiNaC::psi1_eval (
    const ex & x) [static]
```

Evaluation of digamma-function  $\psi(x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1\\_2](#), [\\_ex2](#), [\\_num2\\_p](#), [\\_num1\\_p](#), [Euler](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::inverse\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), and [psi\(\)](#).

**7.1.3.341 psi1\_deriv()**

```
ex GiNaC::psi1_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), and [psi\(\)](#).

**7.1.3.342 psi1\_series()**

```
ex GiNaC::psi1_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [\\_ex1](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.343 psi2\_evalf()**

```
ex GiNaC::psi2_evalf (
    const ex & n,
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [psi\(\)](#).

**7.1.3.344 psi2\_eval()**

```
ex GiNaC::psi2_eval (
    const ex & n,
    const ex & x) [static]
```

Evaluation of polygamma-function  $\psi(n,x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_num1\\_2\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num\\_1\\_p](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [psi\(\)](#), [tgamma\(\)](#), and [zeta\(\)](#).

**7.1.3.345 psi2\_deriv()**

```
ex GiNaC::psi2_deriv (
    const ex & n,
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), and [psi\(\)](#).

**7.1.3.346 psi2\_series()**

```
ex GiNaC::psi2_series (
    const ex & n,
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.347 G2\_evalf()**

```
ex GiNaC::G2_evalf (
    const ex & x_,
    const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [ex\\_to\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), and [pow\(\)](#).

**7.1.3.348 G2\_eval()**

```
ex GiNaC::G2_eval (
    const ex & x_,
    const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), and [pow\(\)](#).

**7.1.3.349 G3\_evalf()**

```
ex GiNaC::G3_evalf (
    const ex & x_,
    const ex & s_,
    const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [GiNaC::info\\_flags::real](#).

**7.1.3.350 G3\_eval()**

```
ex GiNaC::G3_eval (
    const ex & x_,
    const ex & s_,
    const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [GiNaC::info\\_flags::real](#).

**7.1.3.351 Li\_evalf()**

```
ex GiNaC::Li_evalf (
    const ex & m_,
    const ex & x_) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::info\\_flags::posint](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.352 Li\_eval()**

```
ex GiNaC::Li_eval (
    const ex & m_,
    const ex & x_) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_48](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::container< C >::begin\(\)](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), and [zeta\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.353 Li\_series()**

```
ex GiNaC::Li_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.354 Li\_deriv()**

```
ex GiNaC::Li_deriv (
    const ex & m_,
    const ex & x_,
    unsigned deriv_param) [static]
```

References [\\_ex0](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.355 Li\_print\_latex()**

```
void GiNaC::Li_print_latex (
    const ex & m_,
    const ex & x_,
    const print_context & c) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.356 REGISTER\_FUNCTION() [19/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li ,
    evalf_func(Li_evalf). eval_func(Li_eval). series_func(Li_series). derivative_↵
func(Li_deriv). print_func< print_latex >(Li_print_latex). do_not_evalf_params() )
```

References [Li\\_deriv\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [Li\\_print\\_latex\(\)](#), and [Li\\_series\(\)](#).

**7.1.3.357 S\_evalf()**

```
ex GiNaC::S_evalf (
    const ex & n,
    const ex & p,
    const ex & x) [static]
```

References [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), and [GiNaC::info\\_flags::posint](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.358 S\_eval()**

```
ex GiNaC::S_eval (
    const ex & n,
    const ex & p,
    const ex & x) [static]
```

References [\\_ex0](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [to\\_int\(\)](#), and [zeta\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.359 S\_series()**

```
ex GiNaC::S_series (
    const ex & n,
    const ex & p,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.360 S\_deriv()**

```
ex GiNaC::S_deriv (
    const ex & n,
    const ex & p,
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex0](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.361 S\_print\_latex()**

```
void GiNaC::S_print_latex (
    const ex & n,
    const ex & p,
    const ex & x,
    const print_context & c) [static]
```

References [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.362 REGISTER\_FUNCTION()** [20/36]

```
GiNaC::REGISTER_FUNCTION (
    S ,
    evalf_func(S_evalf). eval_func(S_eval). series_func(S_series). derivative_↔
func(S_deriv). print_func< print_latex >(S_print_latex). do_not_evalf_params() )
```

References [S\\_deriv\(\)](#), [S\\_eval\(\)](#), [S\\_evalf\(\)](#), [S\\_print\\_latex\(\)](#), and [S\\_series\(\)](#).

**7.1.3.363 H\_evalf()**

```
ex GiNaC::H_evalf (
    const ex & x1,
    const ex & x2) [static]
```

References [GiNaC::container< C >::append\(\)](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [l](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_a\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::ex::subs\(\)](#), and [to\\_int\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.364 H\_eval()**

```
ex GiNaC::H_eval (
    const ex & m_,
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_a\(\)](#), [log\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), and [step\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.365 H\_series()**

```
ex GiNaC::H_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.366 H\_deriv()**

```
ex GiNaC::H_deriv (
    const ex & m_,
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::container< C >::begin\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), and [GiNaC::container< C >::remove\\_first\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.367 H\_print\_latex()**

```
void GiNaC::H_print_latex (
    const ex & m_,
    const ex & x,
    const print_context & c) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.368 REGISTER\_FUNCTION() [21/36]**

```
GiNaC::REGISTER_FUNCTION (
    H ,
    evalf_func(H_evalf). eval_func(H_eval). series_func(H_series). derivative_↔
func(H_deriv). print_func< print_latex >(H_print_latex). do_not_evalf_params() )
```

References [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [H\\_evalf\(\)](#), [H\\_print\\_latex\(\)](#), and [H\\_series\(\)](#).

**7.1.3.369 zeta1\_evalf()**

```
ex GiNaC::zeta1_evalf (
    const ex & x) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [Digits](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), and [zeta\(\)](#).

Referenced by [zeta1\\_eval\(\)](#).

**7.1.3.370 zeta1\_eval()**

```
ex GiNaC::zeta1_eval (
    const ex & m) [static]
```

References [\\_ex0](#), [\\_ex\\_1\\_2](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [abs\(\)](#), [bernoulli\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::info\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [zeta\(\)](#), and [zeta1\\_evalf\(\)](#).

**7.1.3.371 zeta1\_deriv()**

```
ex GiNaC::zeta1_deriv (
    const ex & m,
    unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), and [is\\_exactly\\_a\(\)](#).

**7.1.3.372 zeta1\_print\_latex()**

```
void GiNaC::zeta1_print_latex (
    const ex & m_,
    const print\_context & c) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

**7.1.3.373 zeta2\_evalf()**

```
ex GiNaC::zeta2_evalf (
    const ex & x,
    const ex & s) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [evalf\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), and [zeta\(\)](#).

**7.1.3.374 zeta2\_eval()**

```
ex GiNaC::zeta2_eval (
    const ex & m,
    const ex & s_) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::info\\_flags::positive](#), and [zeta\(\)](#).

**7.1.3.375 zeta2\_deriv()**

```
ex GiNaC::zeta2_deriv (
    const ex & m,
    const ex & s,
    unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::info\\_flags::positive](#).

**7.1.3.376 zeta2\_print\_latex()**

```
void GiNaC::zeta2_print_latex (
    const ex & m_,
    const ex & s_,
    const print\_context & c) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), and [GiNaC::print\\_context::s](#).

**7.1.3.377 exp\_evalf()**

```
ex GiNaC::exp_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.378 exp\_eval()**

```
ex GiNaC::exp_eval (
    const ex & x) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num3\\_p](#), [\\_num4\\_p](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.379 exp\_expand()**

```
ex GiNaC::exp_expand (
    const ex & arg,
    unsigned options) [static]
```

References [GiNaC::ex::begin\(\)](#), [dynallocate\(\)](#), [GiNaC::ex::end\(\)](#), [exp\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.380 exp\_deriv()**

```
ex GiNaC::exp_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [exp\(\)](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.381 exp\_real\_part()**

```
ex GiNaC::exp_real_part (
    const ex & x) [static]
```

References [cos\(\)](#), [exp\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.382 exp\_imag\_part()**

```
ex GiNaC::exp_imag_part (
    const ex & x) [static]
```

References [exp\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.383 exp\_conjugate()**

```
ex GiNaC::exp_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [exp\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.384 exp\_power()**

```
ex GiNaC::exp_power (
    const ex & x,
    const ex & a) [static]
```

References [\\_ex\\_1](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.385 exp\_info()**

```
bool GiNaC::exp_info (
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.386 REGISTER\_FUNCTION() [22/36]**

```
GiNaC::REGISTER_FUNCTION (
    exp ,
    eval_func(exp_eval). evalf_func(exp_evalf). info_func(exp_info). expand_↔
func(exp_expand). derivative_func(exp_deriv). real_part_func(exp_real_part). imag_part_↔
func(exp_imag_part). conjugate_func(exp_conjugate). power_func(exp_power). latex_name("\\exp")
)
```

References [exp\(\)](#), [exp\\_conjugate\(\)](#), [exp\\_deriv\(\)](#), [exp\\_eval\(\)](#), [exp\\_evalf\(\)](#), [exp\\_expand\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_info\(\)](#), [exp\\_power\(\)](#), and [exp\\_real\\_part\(\)](#).

**7.1.3.387 log\_evalf()**

```
ex GiNaC::log_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [log\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.388 log\_eval()**

```
ex GiNaC::log_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1\\_2](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::rational](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.389 log\_deriv()**

```
ex GiNaC::log_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex\\_1](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.390 log\_series()**

```
ex GiNaC::log_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::pseries::add\\_series\(\)](#), [coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [csgn\(\)](#), [GiNaC::ex::diff\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::pseries::ldegree\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::pseries::nops\(\)](#), [Pi](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.391 log\_real\_part()**

```
ex GiNaC::log_real_part (
    const ex & x) [static]
```

References [abs\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), and [GiNaC::info\\_flags::nonnegative](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.392 log\_imag\_part()**

```
ex GiNaC::log_imag_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), and [real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.393 log\_expand()**

```
ex GiNaC::log_expand (
    const ex & arg,
    unsigned options) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.394 log\_conjugate()**

```
ex GiNaC::log_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), and [GiNaC::info\\_flags::positive](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.395 log\_info()**

```
bool GiNaC::log_info (
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::positive](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.396 REGISTER\_FUNCTION() [23/36]**

```
GiNaC::REGISTER_FUNCTION (
    log ,
    eval_func(log_eval). evalf_func(log_evalf). info_func(log_info). expand_↔
func(log_expand). derivative_func(log_deriv). series_func(log_series). real_part_func(log_real_part).
imag_part_func(log_imag_part). conjugate_func(log_conjugate). latex_name("\\ln" )
```

References [log\(\)](#), [log\\_conjugate\(\)](#), [log\\_deriv\(\)](#), [log\\_eval\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [log\\_imag\\_part\(\)](#), [log\\_info\(\)](#), [log\\_real\\_part\(\)](#), and [log\\_series\(\)](#).

**7.1.3.397 sin\_evalf()**

```
ex GiNaC::sin_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.398 sin\_eval()**

```
ex GiNaC::sin_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num15\\_p](#), [\\_num18\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [\\_num6\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sin\(\)](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.399 sin\_deriv()**

```
ex GiNaC::sin_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [cos\(\)](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.400 sin\_real\_part()**

```
ex GiNaC::sin_real_part (
    const ex & x) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.401 sin\_imag\_part()**

```
ex GiNaC::sin_imag_part (
    const ex & x) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.402 sin\_conjugate()**

```
ex GiNaC::sin_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.403 trig\_info()**

```
bool GiNaC::trig_info (
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), [REGISTER\\_FUNCTION\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.404 REGISTER\_FUNCTION() [24/36]**

```
GiNaC::REGISTER_FUNCTION (
    sin ,
    eval_func(sin\_eval) . evalf_func(sin\_evalf) . info_func(trig\_info) . derivative_↵
func(sin\_deriv) . real_part_func(sin\_real\_part) . imag_part_func(sin\_imag\_part) . conjugate_↵
func(sin\_conjugate) . latex_name("\\sin" ) )
```

References [sin\(\)](#), [sin\\_conjugate\(\)](#), [sin\\_deriv\(\)](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), and [trig\\_info\(\)](#).

**7.1.3.405 cos\_evalf()**

```
ex GiNaC::cos_evalf (
    const ex & x) [static]
```

References [cos\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.406 cos\_eval()**

```
ex GiNaC::cos_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num12\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num24\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [cos\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.407 cos\_deriv()**

```
ex GiNaC::cos_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.408 cos\_real\_part()**

```
ex GiNaC::cos_real_part (
    const ex & x) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.409 cos\_imag\_part()**

```
ex GiNaC::cos_imag_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.410 cos\_conjugate()**

```
ex GiNaC::cos_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [cos\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.411 REGISTER\_FUNCTION() [25/36]**

```
GiNaC::REGISTER_FUNCTION (
    cos ,
    eval_func(cos_eval). info_func(trig_info). evalf_func(cos_evalf). derivative_↔
func(cos_deriv). real_part_func(cos_real_part). imag_part_func(cos_imag_part). conjugate_↔
func(cos_conjugate). latex_name("\\cos") )
```

References [cos\(\)](#), [cos\\_conjugate\(\)](#), [cos\\_deriv\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), and [trig\\_info\(\)](#).

**7.1.3.412 tan\_evalf()**

```
ex GiNaC::tan_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [tan\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.413 tan\_eval()**

```
ex GiNaC::tan_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_3](#), [\\_ex2](#), [\\_ex3](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sqrt\(\)](#), and [tan\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.414 tan\_deriv()**

```
ex GiNaC::tan_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), and [tan\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.415 tan\_real\_part()**

```
ex GiNaC::tan_real_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), and [tan\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.416 tan\_imag\_part()**

```
ex GiNaC::tan_imag_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.417 tan\_series()**

```
ex GiNaC::tan_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [cos\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [Pi](#), [sin\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.418 tan\_conjugate()**

```
ex GiNaC::tan_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [tan\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.419 REGISTER\_FUNCTION() [26/36]**

```
GiNaC::REGISTER_FUNCTION (
    tan ,
    eval_func(tan_eval). evalf_func(tan_evalf). info_func(trig_info). derivative←
_func(tan_deriv). series_func(tan_series). real_part_func(tan_real_part). imag_part←
func(tan_imag_part). conjugate_func(tan_conjugate). latex_name("\\tan" )
```

References [tan\(\)](#), [tan\\_conjugate\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tan\\_series\(\)](#), and [trig\\_info\(\)](#).

**7.1.3.420 asin\_evalf()**

```
ex GiNaC::asin_evalf (
    const ex & x) [static]
```

References [asin\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.421 asin\_eval()**

```
ex GiNaC::asin_eval (
    const ex & x) [static]
```

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [asin\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.422 asin\_deriv()**

```
ex GiNaC::asin_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.423 asin\_conjugate()**

```
ex GiNaC::asin_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asin\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.424 asin\_info()**

```
bool GiNaC::asin_info (
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), and [GiNaC::ex::info\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [REGISTER\\_FUNCTION\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.425 REGISTER\_FUNCTION() [27/36]**

```
GiNaC::REGISTER_FUNCTION (
    asin ,
    eval_func(asin_eval). evalf_func(asin_evalf). info_func(asin_info). derivative↔
_func(asin_deriv). conjugate_func(asin_conjugate). latex_name("\\arcsin") )
```

References [asin\(\)](#), [asin\\_conjugate\(\)](#), [asin\\_deriv\(\)](#), [asin\\_eval\(\)](#), [asin\\_evalf\(\)](#), and [asin\\_info\(\)](#).

**7.1.3.426 acos\_evalf()**

```
ex GiNaC::acos_evalf (
    const ex & x) [static]
```

References [acos\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.427 acos\_eval()**

```
ex GiNaC::acos_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acos\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.428 acos\_deriv()**

```
ex GiNaC::acos_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.429 acos\_conjugate()**

```
ex GiNaC::acos_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [acos\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.430 REGISTER\_FUNCTION() [28/36]**

```
GiNaC::REGISTER_FUNCTION (
    acos ,
    eval_func(acos_eval) . evalf_func(acos_evalf) . info_func(asin_info) . derivative↔
    _func(acos_deriv) . conjugate_func(acos_conjugate) . latex_name("\\arccos") )
```

References [acos\(\)](#), [acos\\_conjugate\(\)](#), [acos\\_deriv\(\)](#), [acos\\_eval\(\)](#), [acos\\_evalf\(\)](#), and [asin\\_info\(\)](#).

**7.1.3.431 atan\_evalf()**

```
ex GiNaC::atan_evalf (
    const ex & x) [static]
```

References [atan\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.432 atan\_eval()**

```
ex GiNaC::atan_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_4](#), [\\_ex\\_1](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.433 atan\_deriv()**

```
ex GiNaC::atan_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.434 atan\_series()**

```
ex GiNaC::atan_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atan\(\)](#), [csgn\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [I](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.435 atan\_conjugate()**

```
ex GiNaC::atan_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atan\(\)](#), [GiNaC::ex::conjugate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), and [GiNaC::ex::real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.436 atan\_info()**

```
bool GiNaC::atan_info (
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [REGISTER\\_FUNCTION\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.437 REGISTER\_FUNCTION() [29/36]**

```
GiNaC::REGISTER_FUNCTION (
    atan ,
    eval_func(atan_eval). evalf_func(atan_evalf). info_func(atan_info). derivative↔
_func(atan_deriv). series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\\arctan")
)
```

References [atan\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_evalf\(\)](#), [atan\\_info\(\)](#), and [atan\\_series\(\)](#).

**7.1.3.438 atan2\_evalf()**

```
ex GiNaC::atan2_evalf (
    const ex & y,
    const ex & x) [static]
```

References [atan\(\)](#), [ex\\_to\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.439 atan2\_eval()**

```
ex GiNaC::atan2_eval (
    const ex & y,
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1\\_2](#), [\\_ex1\\_4](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [Pi](#), [GiNaC::info\\_flags::positive](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.440 atan2\_deriv()**

```
ex GiNaC::atan2_deriv (
    const ex & y,
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.441 atan2\_info()**

```
bool GiNaC::atan2_info (
    const ex & y,
    const ex & x,
    unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), and [GiNaC::info\\_flags::real](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.442 REGISTER\_FUNCTION() [30/36]**

```
GiNaC::REGISTER_FUNCTION (
    atan2 ,
    eval_func(atan2_eval). evalf_func(atan2_evalf). info_func(atan2_info). evalf_←
func(atan2_evalf). derivative_func(atan2_deriv) )
```

References [atan2\\_deriv\(\)](#), [atan2\\_eval\(\)](#), [atan2\\_evalf\(\)](#), and [atan2\\_info\(\)](#).

#### 7.1.3.443 `sinh_evalf()`

```
ex GiNaC::sinh_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.444 `sinh_eval()`

```
ex GiNaC::sinh_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sin\(\)](#), [sinh\(\)](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.445 `sinh_deriv()`

```
ex GiNaC::sinh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [cosh\(\)](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.446 `sinh_real_part()`

```
ex GiNaC::sinh_real_part (
    const ex & x) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

#### 7.1.3.447 `sinh_imag_part()`

```
ex GiNaC::sinh_imag_part (
    const ex & x) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [sin\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.448 sinh\_conjugate()**

```
ex GiNaC::sinh_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.449 REGISTER\_FUNCTION() [31/36]**

```
GiNaC::REGISTER_FUNCTION (
    sinh ,
    eval_func(sinh_eval). evalf_func(sinh_evalf). info_func(atan_info). derivative↔
_func(sinh_deriv). real_part_func(sinh_real_part). imag_part_func(sinh_imag_part). conjugate↔
_func(sinh_conjugate). latex_name("\\sinh") )
```

References [atan\\_info\(\)](#), [sinh\(\)](#), [sinh\\_conjugate\(\)](#), [sinh\\_deriv\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [sinh\\_imag\\_part\(\)](#), and [sinh\\_real\\_part\(\)](#).

**7.1.3.450 cosh\_evalf()**

```
ex GiNaC::cosh_evalf (
    const ex & x) [static]
```

References [cosh\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.451 cosh\_eval()**

```
ex GiNaC::cosh_eval (
    const ex & x) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [cos\(\)](#), [cosh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), and [sqrt\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.452 cosh\_deriv()**

```
ex GiNaC::cosh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.453 cosh\_real\_part()**

```
ex GiNaC::cosh_real_part (
    const ex & x) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.454 cosh\_imag\_part()**

```
ex GiNaC::cosh_imag_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [sinh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.455 cosh\_conjugate()**

```
ex GiNaC::cosh_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [cosh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.456 REGISTER\_FUNCTION() [32/36]**

```
GiNaC::REGISTER_FUNCTION (
    cosh ,
    eval_func(cosh_eval). evalf_func(cosh_evalf). info_func(exp_info). derivative↔
_func(cosh_deriv). real_part_func(cosh_real_part). imag_part_func(cosh_imag_part). conjugate↔
_func(cosh_conjugate). latex_name("\\cosh") )
```

References [cosh\(\)](#), [cosh\\_conjugate\(\)](#), [cosh\\_deriv\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), and [exp\\_info\(\)](#).

**7.1.3.457 tanh\_evalf()**

```
ex GiNaC::tanh_evalf (
    const ex & x) [static]
```

References [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_exactly\\_a\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.458 tanh\_eval()**

```
ex GiNaC::tanh_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sqrt\(\)](#), [tan\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.459 tanh\_deriv()**

```
ex GiNaC::tanh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.460 tanh\_series()**

```
ex GiNaC::tanh_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [cosh\(\)](#), [GINAC\\_ASSERT](#), [I](#), [is\\_a\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [Pi](#), [sinh\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.461 tanh\_real\_part()**

```
ex GiNaC::tanh_real_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.462 tanh\_imag\_part()**

```
ex GiNaC::tanh_imag_part (
    const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.463 tanh\_conjugate()**

```
ex GiNaC::tanh_conjugate (
    const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), and [tanh\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.464 REGISTER\_FUNCTION() [33/36]**

```
GiNaC::REGISTER_FUNCTION (
    tanh ,
    eval_func(tanh_eval). evalf_func(tanh_evalf). info_func(atan_info). derivative↔
    _func(tanh_deriv). series_func(tanh_series). real_part_func(tanh_real_part). imag_part_↔
    func(tanh_imag_part). conjugate_func(tanh_conjugate). latex_name("\\tanh") )
```

References [atan\\_info\(\)](#), [tanh\(\)](#), [tanh\\_conjugate\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tanh\\_imag\\_part\(\)](#), [tanh\\_real\\_part\(\)](#), and [tanh\\_series\(\)](#).

**7.1.3.465 asinh\_evalf()**

```
ex GiNaC::asinh_evalf (
    const ex & x) [static]
```

References [asinh\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.466 asinh\_eval()**

```
ex GiNaC::asinh_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [asinh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), and [GiNaC::info\\_flags::numeric](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.467 asinh\_deriv()**

```
ex GiNaC::asinh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.468 asinh\_conjugate()**

```
ex GiNaC::asinh_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asinh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), and [GiNaC::ex::real\\_part\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.469 REGISTER\_FUNCTION() [34/36]**

```
GiNaC::REGISTER_FUNCTION (
    asinh ,
    eval_func (asinh_eval) . evalf_func (asinh_evalf) . info_func (atan_info) . derivative←
_func (asinh_deriv) . conjugate_func (asinh_conjugate) )
```

References [asinh\(\)](#), [asinh\\_conjugate\(\)](#), [asinh\\_deriv\(\)](#), [asinh\\_eval\(\)](#), [asinh\\_evalf\(\)](#), and [atan\\_info\(\)](#).

**7.1.3.470 acosh\_evalf()**

```
ex GiNaC::acosh_evalf (
    const ex & x) [static]
```

References [acosh\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.471 acosh\_eval()**

```
ex GiNaC::acosh_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [acosh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.472 acosh\_deriv()**

```
ex GiNaC::acosh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.473 `acosh_conjugate()`**

```
ex GiNaC::acosh_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [acosh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.474 `REGISTER_FUNCTION()` [35/36]**

```
GiNaC::REGISTER_FUNCTION (
    acosh ,
    eval_func(acosh_eval) . evalf_func(acosh_evalf) . info_func(asin_info) . derivative←
_func(acosh_deriv) . conjugate_func(acosh_conjugate) )
```

References [acosh\(\)](#), [acosh\\_conjugate\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [acosh\\_evalf\(\)](#), and [asin\\_info\(\)](#).

**7.1.3.475 `atanh_evalf()`**

```
ex GiNaC::atanh_evalf (
    const ex & x) [static]
```

References [atanh\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), and [is\\_exactly\\_a\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.476 `atanh_eval()`**

```
ex GiNaC::atanh_eval (
    const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), and [GiNaC::info\\_flags::numeric](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.477 `atanh_deriv()`**

```
ex GiNaC::atanh_deriv (
    const ex & x,
    unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), and [GINAC\\_ASSERT](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.478 atanh\_series()**

```
ex GiNaC::atanh_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atanh\(\)](#), [csgn\(\)](#), [ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [I](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.479 atanh\_conjugate()**

```
ex GiNaC::atanh_conjugate (
    const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atanh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [is\\_exactly\\_a\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.480 REGISTER\_FUNCTION() [36/36]**

```
GiNaC::REGISTER_FUNCTION (
    atanh ,
    eval_func(atanh_eval). evalf_func(atanh_evalf). info_func(asin_info). derivative←
_func(atanh_deriv). series_func(atanh_series). conjugate_func(atanh_conjugate) )
```

References [asin\\_info\(\)](#), [atanh\(\)](#), [atanh\\_conjugate\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_evalf\(\)](#), and [atanh\\_series\(\)](#).

**7.1.3.481 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [12/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integral ,
    basic ,
    print_func< print_dflt > &::do_print. print_func< print_python > &::do_print.
print_func< print_latex > &::do_print_latex )
```

References [GiNaC::integral::do\\_print\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), and [dynalocate\(\)](#).

**7.1.3.482 subsvalue()**

```
ex GiNaC::subsvalue (
    const ex & var,
    const ex & value,
    const ex & fun)
```

References [GiNaC::ex::evalf\(\)](#), [is\\_a\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [adaptivesimpson\(\)](#).

**7.1.3.483 adaptivesimpson()**

```
GiNaC::ex GiNaC::adaptivesimpson (
    const ex & x,
    const ex & a_in,
    const ex & b_in,
    const ex & f,
    const ex & error)
```

Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.

Parameters are integration variable, left boundary, right boundary, function to be integrated and the relative integration error. The function should evalf into a number after substituting the integration variable by a number. Another thing to note is that this implementation is no good at integrating functions with discontinuities.

References [abs\(\)](#), [GiNaC::ex::evalf\(\)](#), [ex\\_to\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::integral::max\\_integration\\_level](#), [GiNaC::ex::subs\(\)](#), and [subsvalue\(\)](#).

Referenced by [GiNaC::integral::evalf\(\)](#).

**7.1.3.484 GINAC\_BIND\_UNARCHIVER()** [22/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integral )
```

**7.1.3.485 GINAC\_DECLARE\_UNARCHIVER()** [23/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integral )
```

References [GiNaC::integral::relative\\_integration\\_error](#).

**7.1.3.486 ifactor()**

```
ex GiNaC::ifactor (
    const numeric & n)
```

Returns the decomposition of the positive integer n into prime numbers in the form  $\text{lst}(p_1, \dots, p_r), \text{lst}(a_1, \dots, a_r)$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .

References [GiNaC::container< C >::append\(\)](#), [irem\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), and [GiNaC::info\\_flags::prime](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), and [kronecker\\_symbol\(\)](#).

**7.1.3.487 is\_discriminant\_of\_quadratic\_number\_field()**

```
bool GiNaC::is_discriminant_of_quadratic_number_field (
    const numeric & n)
```

Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.

Returns false otherwise.

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [abs\(\)](#), [ex\\_to\(\)](#), [ifactor\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [mod\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#).

**7.1.3.488 kronecker\_symbol()**

```
numeric GiNaC::kronecker_symbol (
    const numeric & a,
    const numeric & n)
```

Returns the Kronecker symbol  $a$ : integer  $n$ : integer.

This routine defines  $\text{kronecker\_symbol}(1,0) = 1$   $\text{kronecker\_symbol}(-1,0) = 1$   $\text{kronecker\_symbol}(a,0) = 0$ ,  $a \neq 1, -1$

In particular  $\text{kronecker\_symbol}(-1,0) = 1$  (in agreement with Sage)

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\\_to\(\)](#), [ifactor\(\)](#), [GiNaC::numeric::is\\_even\(\)](#), [GiNaC::container< C >::op\(\)](#), and [pow\(\)](#).

Referenced by [primitive\\_dirichlet\\_character\(\)](#).

**7.1.3.489 primitive\_dirichlet\_character()**

```
numeric GiNaC::primitive_dirichlet_character (
    const numeric & n,
    const numeric & a)
```

Defines a primitive Dirichlet character through the Kronecker symbol.

$n$ : integer  $a$ : discriminant of a quadratic field  $|a|$ : conductor

The character takes the values  $-1, 0, 1$ .

References [kronecker\\_symbol\(\)](#).

Referenced by [dirichlet\\_character\(\)](#), and [generalised\\_Bernoulli\\_number\(\)](#).

**7.1.3.490 `dirichlet_character()`**

```
numeric GiNaC::dirichlet_character (
    const numeric & n,
    const numeric & a,
    const numeric & N)
```

Defines a Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor N: modulus, needs to be multiple of  $|a|$

The character takes the values -1,0,1.

References [gcd\(\)](#), and [primitive\\_dirichlet\\_character\(\)](#).

**7.1.3.491 `generalised_Bernoulli_number()`**

```
numeric GiNaC::generalised_Bernoulli_number (
    const numeric & k,
    const numeric & b)
```

The generalised Bernoulli number.

k: index / modular weight

b: discriminant of a quadratic field, defines primitive character  $\psi$   $M=|b|$ : conductor of primitive character  $\psi$

The generalised Bernoulli number is computed from the series expansion of the generating function. The generating function is given in eq.(34), arXiv:1704.08895

References [abs\(\)](#), [GiNaC::ex::coeff\(\)](#), [ex\\_to\(\)](#), [exp\(\)](#), [factorial\(\)](#), [primitive\\_dirichlet\\_character\(\)](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

**7.1.3.492 `Bernoulli_polynomial()`**

```
ex GiNaC::Bernoulli_polynomial (
    const numeric & k,
    const ex & x)
```

The Bernoulli polynomials.

References [GiNaC::ex::coeff\(\)](#), [exp\(\)](#), [factorial\(\)](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#).

**7.1.3.493 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [13/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integration_kernel ,
    basic ,
    print_func< print_context > &::do_print )
```

References [GiNaC::integration\\_kernel::do\\_print\(\)](#).

**7.1.3.494 GINAC\_BIND\_UNARCHIVER()** [23/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integration_kernel )
```

**7.1.3.495 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [14/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic_log_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [GiNaC::basic\\_log\\_kernel::do\\_print\(\)](#).

**7.1.3.496 GINAC\_BIND\_UNARCHIVER()** [24/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    basic_log_kernel )
```

**7.1.3.497 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [15/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    multiple_polylog_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex1](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), and [GiNaC::multiple\\_polylog\\_kernel::multiple\\_polylog\\_kernel\(\)](#).

**7.1.3.498 GINAC\_BIND\_UNARCHIVER()** [25/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    multiple_polylog_kernel )
```

**7.1.3.499 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [16/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ELi_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), and [GiNaC::ELi\\_kernel::do\\_print\(\)](#).

**7.1.3.500 GINAC\_BIND\_UNARCHIVER()** [26/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ELi_kernel )
```

**7.1.3.501 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [17/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Ebar_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), and [GiNaC::Ebar\\_kernel::do\\_print\(\)](#).

**7.1.3.502 GINAC\_BIND\_UNARCHIVER()** [27/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Ebar_kernel )
```

**7.1.3.503 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [18/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dtau_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), [\\_ex1](#), and [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#).

**7.1.3.504 GINAC\_BIND\_UNARCHIVER()** [28/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**7.1.3.505 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [19/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dz_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), [\\_ex1](#), and [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#).

**7.1.3.506 GINAC\_BIND\_UNARCHIVER()** [29/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dz_kernel )
```

**7.1.3.507 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [20/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), [\\_ex1](#), and [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#).

**7.1.3.508 GINAC\_BIND\_UNARCHIVER()** [30/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_kernel )
```

**7.1.3.509 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [21/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_h_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), [\\_ex1](#), and [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#).

**7.1.3.510 GINAC\_BIND\_UNARCHIVER()** [31/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_h_kernel )
```

**7.1.3.511 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [22/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    modular_form_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), [\\_ex1](#), and [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#).

**7.1.3.512 GINAC\_BIND\_UNARCHIVER()** [32/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    modular_form_kernel )
```

**7.1.3.513 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [23/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    user_defined_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex0](#), and [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#).

**7.1.3.514 GINAC\_BIND\_UNARCHIVER()** [33/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    user_defined_kernel )
```



**7.1.3.515 GINAC\_DECLARE\_UNARCHIVER()** [24/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integration_kernel )
```

**7.1.3.516 GINAC\_DECLARE\_UNARCHIVER()** [25/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    basic_log_kernel )
```

**7.1.3.517 GINAC\_DECLARE\_UNARCHIVER()** [26/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    multiple_polylog_kernel )
```

**7.1.3.518 GINAC\_DECLARE\_UNARCHIVER()** [27/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ELi_kernel )
```

**7.1.3.519 GINAC\_DECLARE\_UNARCHIVER()** [28/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Ebar_kernel )
```

**7.1.3.520 GINAC\_DECLARE\_UNARCHIVER()** [29/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**7.1.3.521 GINAC\_DECLARE\_UNARCHIVER()** [30/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dz_kernel )
```

**7.1.3.522 GINAC\_DECLARE\_UNARCHIVER()** [31/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_kernel )
```

**7.1.3.523 GINAC\_DECLARE\_UNARCHIVER()** [32/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_h_kernel )
```

**7.1.3.524 GINAC\_DECLARE\_UNARCHIVER()** [33/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    modular_form_kernel )
```

**7.1.3.525 GINAC\_DECLARE\_UNARCHIVER()** [34/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    user_defined_kernel )
```

**7.1.3.526 GINAC\_DECLARE\_UNARCHIVER()** [35/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    lst )
```

**7.1.3.527 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [24/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    matrix ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr >
&::do_print_python_repr )
```

Default ctor.

Initializes to 1 x 1-dimensional zero-matrix.

References [\\_ex0](#), [GiNaC::matrix::do\\_print\(\)](#), [GiNaC::matrix::do\\_print\\_latex\(\)](#), [GiNaC::matrix::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), and [GiNaC::status\\_flags::not\\_shareable](#).

**7.1.3.528 GINAC\_BIND\_UNARCHIVER()** [34/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    matrix )
```

**7.1.3.529 lst\_to\_matrix()**

```
ex GiNaC::lst_to_matrix (
    const lst & l)
```

Convert list of lists to matrix.

References [cols\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [rows\(\)](#).

**7.1.3.530 diag\_matrix()** [1/2]

```
ex GiNaC::diag_matrix (
    const lst & l)
```

Convert list of diagonal elements to matrix.

References [dynallocate\(\)](#), and [GiNaC::container< C >::nops\(\)](#).

**7.1.3.531 diag\_matrix()** [2/2]

```
ex GiNaC::diag_matrix (
    std::initializer_list< ex > l)
```

References [dynallocate\(\)](#).

**7.1.3.532 unit\_matrix()** [1/2]

```
ex GiNaC::unit_matrix (
    unsigned r,
    unsigned c)
```

Create an r times c unit matrix.

References [\\_ex1](#), [dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [unit\\_matrix\(\)](#).

**7.1.3.533 symbolic\_matrix()** [1/2]

```
ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name,
    const std::string & tex_base_name)
```

Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/ $\leftrightarrow$  column position of each element.

The base name for LaTeX output is specified separately.

References [dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [symbolic\\_matrix\(\)](#).

**7.1.3.534 reduced\_matrix()**

```
ex GiNaC::reduced_matrix (
    const matrix & m,
    unsigned r,
    unsigned c)
```

Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.

The determinant of the result is the Minor r, c.

References [cols\(\)](#), [GiNaC::matrix::cols\(\)](#), [dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::matrix::rows\(\)](#), [rows\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**7.1.3.535 sub\_matrix()**

```
ex GiNaC::sub_matrix (
    const matrix & m,
    unsigned r,
    unsigned nr,
    unsigned c,
    unsigned nc)
```

Return the nr times nc submatrix starting at position r, c of matrix m.

References [GiNaC::matrix::cols\(\)](#), [dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::matrix::rows\(\)](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [lst\\_to\\_clifford\(\)](#).

**7.1.3.536 GINAC\_DECLARE\_UNARCHIVER()** [36/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    matrix )
```

**7.1.3.537 nops()** [2/2]

```
size_t GiNaC::nops (
    const matrix & m) [inline]
```

References [GiNaC::matrix::nops\(\)](#).

**7.1.3.538 expand()** [2/2]

```
ex GiNaC::expand (
    const matrix & m,
    unsigned options = 0) [inline]
```

References [GiNaC::basic::expand\(\)](#).

**7.1.3.539 evalf()** [2/2]

```
ex GiNaC::evalf (
    const matrix & m) [inline]
```

References [GiNaC::basic::evalf\(\)](#).

**7.1.3.540 rows()**

```
unsigned GiNaC::rows (
    const matrix & m) [inline]
```

References [GiNaC::matrix::rows\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#), [clifford\\_moebius\\_map\(\)](#), [lst\\_to\\_clifford\(\)](#), [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

**7.1.3.541 cols()**

```
unsigned GiNaC::cols (
    const matrix & m) [inline]
```

References [GiNaC::matrix::cols\(\)](#).

Referenced by [clifford\\_moebius\\_map\(\)](#), [clifford\\_moebius\\_map\(\)](#), [lst\\_to\\_clifford\(\)](#), [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

**7.1.3.542 transpose()**

```
matrix GiNaC::transpose (
    const matrix & m) [inline]
```

References [GiNaC::matrix::transpose\(\)](#).

**7.1.3.543 determinant()**

```
ex GiNaC::determinant (
    const matrix & m,
    unsigned options = determinant\_algo::automatic) [inline]
```

References [GiNaC::determinant\\_algo::automatic](#), and [GiNaC::matrix::determinant\(\)](#).

**7.1.3.544 trace()**

```
ex GiNaC::trace (
    const matrix & m) [inline]
```

References [GiNaC::matrix::trace\(\)](#).

#### 7.1.3.545 charpoly()

```
ex GiNaC::charpoly (
    const matrix & m,
    const ex & lambda) [inline]
```

References [GiNaC::matrix::charpoly\(\)](#).

#### 7.1.3.546 inverse() [1/3]

```
matrix GiNaC::inverse (
    const matrix & m) [inline]
```

References [GiNaC::solve\\_algo::automatic](#), and [GiNaC::matrix::inverse\(\)](#).

#### 7.1.3.547 inverse() [2/3]

```
matrix GiNaC::inverse (
    const matrix & m,
    unsigned algo) [inline]
```

References [GiNaC::matrix::inverse\(\)](#).

#### 7.1.3.548 rank() [1/2]

```
unsigned GiNaC::rank (
    const matrix & m) [inline]
```

References [GiNaC::matrix::rank\(\)](#).

#### 7.1.3.549 rank() [2/2]

```
unsigned GiNaC::rank (
    const matrix & m,
    unsigned solve_algo) [inline]
```

References [GiNaC::matrix::rank\(\)](#).

#### 7.1.3.550 unit\_matrix() [2/2]

```
ex GiNaC::unit_matrix (
    unsigned x) [inline]
```

Create a x times x unit matrix.

References [unit\\_matrix\(\)](#).

**7.1.3.551 symbolic\_matrix()** [2/2]

```
ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name) [inline]
```

Create an  $r$  times  $c$  matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

References [symbolic\\_matrix\(\)](#).

**7.1.3.552 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [25/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    mul ,
    expairseq ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )
```

References [GiNaC::mul::do\\_print\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), and [GiNaC::basic::do\\_print\\_tree\(\)](#).

**7.1.3.553 tryfactsubs()**

```
bool GiNaC::tryfactsubs (
    const ex & origfactor,
    const ex & patternfactor,
    int & nummatches,
    exmap & repls)
```

References [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::match\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::power::subs\(\)](#).

**7.1.3.554 algebraic\_match\_mul\_with\_mul()**

```
bool GiNaC::algebraic_match_mul_with_mul (
    const mul & e,
    const ex & pat,
    exmap & repls,
    int factor,
    int & nummatches,
    const std::vector< bool > & substed,
    std::vector< bool > & matched)
```

Checks whether  $e$  matches to the pattern  $pat$  and the (possibly to be updated) list of replacements  $repls$ .

This matching is in the sense of algebraic substitutions. Matching starts with  $pat.op(factor)$  of the pattern because the factors before this one have already been matched. The (possibly updated) number of matches is in  $nummatches$ .  $substed[i]$  is true for factors that already have been replaced by previous substitutions and  $matched[i]$  is true for factors that have been matched by the current match.

References [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::op\(\)](#), and [tryfactsubs\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::mul::has\(\)](#).

**7.1.3.555 GINAC\_BIND\_UNARCHIVER()** [35/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    mul )
```

**7.1.3.556 GINAC\_DECLARE\_UNARCHIVER()** [37/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    mul )
```

**7.1.3.557 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [26/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ncmul ,
    exprseq ,
    print_func< print_context > &::do_print, print_func< print_tree > &::do_print↵
    _tree, print_func< print_csrc > &::do_print_csrc, print_func< print_python_repr > &::do_↵
    print_csrc )
```

References [GiNaC::ncmul::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\\_csrc\(\)](#), and [GiNaC::basic::do\\_print\\_tree\(\)](#).

**7.1.3.558 reeval\_ncmul()**

```
ex GiNaC::reeval_ncmul (
    const exvector & v)
```

Referenced by [GiNaC::clifford::eval\\_ncmul\(\)](#).

**7.1.3.559 hold\_ncmul()**

```
ex GiNaC::hold_ncmul (
    const exvector & v)
```

Referenced by [GiNaC::basic::eval\\_ncmul\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::color::eval\\_ncmul\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::eval\\_ncmul\(\)](#).

**7.1.3.560 GINAC\_BIND\_UNARCHIVER()** [36/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ncmul )
```

**7.1.3.561 GINAC\_DECLARE\_UNARCHIVER()** [38/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ncmul )
```



### 7.1.3.562 `get_first_symbol()`

```
bool GiNaC::get_first_symbol (
    const ex & e,
    ex & x) [static]
```

Return pointer to first symbol found in expression.

Due to [GiNaC](#)'s internal ordering of terms, it may not be obvious which symbol this function returns for a given expression.

#### Parameters

<i>e</i>	expression to search
----------	----------------------

<code>x</code>	first symbol found (returned)
----------------	-------------------------------

#### Returns

"false" if no symbol was found, "true" otherwise

References [get\\_first\\_symbol\(\)](#), [is\\_a\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [divide\(\)](#), [frac\\_cancel\(\)](#), [get\\_first\\_symbol\(\)](#), and [GiNaC::ex::unit\(\)](#).

#### 7.1.3.563 add\_symbol()

```
void GiNaC::add_symbol (
    const ex & s,
    sym\_desc\_vec & v) [static]
```

Referenced by [collect\\_symbols\(\)](#).

#### 7.1.3.564 collect\_symbols()

```
void GiNaC::collect_symbols (
    const ex & e,
    sym\_desc\_vec & v) [static]
```

References [add\\_symbol\(\)](#), [collect\\_symbols\(\)](#), [is\\_a\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [collect\\_symbols\(\)](#), and [get\\_symbol\\_stats\(\)](#).

#### 7.1.3.565 get\_symbol\_stats()

```
void GiNaC::get_symbol_stats (
    const ex & a,
    const ex & b,
    sym\_desc\_vec & v) [static]
```

Collect statistical information about symbols in polynomials.

This function fills in a vector of "sym\_desc" structs which contain information about the highest and lowest degrees of all symbols that appear in two polynomials. The vector is then sorted by minimum degree (lowest to highest). The information gathered by this function is used by the GCD routines to identify trivial factors and to determine which variable to choose as the main variable for GCD computation.

#### Parameters

<code>a</code>	first multivariate polynomial
<code>b</code>	second multivariate polynomial
<code>v</code>	vector of <a href="#">sym_desc</a> structs (filled in)

References [collect\\_symbols\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [divide\\_in\\_z\(\)](#), [gcd\(\)](#), and [sqrfree\(\)](#).

**7.1.3.566 lcmcoeff()**

```
numeric GiNaC::lcmcoeff (
    const ex & e,
    const numeric & l) [static]
```

References [\\_num1\\_p](#), [denom\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_a\(\)](#), [is\\_exactly\\_a\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational](#).

Referenced by [heur\\_gcd\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), and [multiply\\_lcm\(\)](#).

**7.1.3.567 lcm\_of\_coefficients\_denominators()**

```
numeric GiNaC::lcm_of_coefficients_denominators (
    const ex & e) [static]
```

Compute LCM of denominators of coefficients of a polynomial.

Given a polynomial with rational coefficients, this function computes the LCM of the denominators of all coefficients. This can be used to bring a polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$ .

**Parameters**

<i>e</i>	multivariate polynomial (need not be expanded)
----------	--

**Returns**

LCM of denominators of coefficients

References [\\_num1\\_p](#), and [lcmcoeff\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [heur\\_gcd\(\)](#), and [sqrfree\(\)](#).

**7.1.3.568 multiply\_lcm()**

```
ex GiNaC::multiply_lcm (
    const ex & e,
    const numeric & lcm) [static]
```

Bring polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.

**Parameters**

<i>e</i>	multivariate polynomial (need not be expanded)
<i>lcm</i>	LCM to multiply in

References [\\_num1\\_p](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [is\\_a\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).

### 7.1.3.569 quo()

```
ex GiNaC::quo (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args)
```

Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

#### Parameters

<i>a</i>	first polynomial in x (dividend)
<i>b</i>	second polynomial in x (divisor)
<i>x</i>	a and b are polynomials in x
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

#### Returns

quotient of a and b in  $\mathbb{Q}[x]$

References [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [dynallocate\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::ex::primpart\(\)](#), [sqrfree\\_parfrac\(\)](#), [sqrfree\\_yun\(\)](#), and [GiNaC::ex::unitcontprim\(\)](#).

### 7.1.3.570 rem()

```
ex GiNaC::rem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args)
```

Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

#### Parameters

<i>a</i>	first polynomial in x (dividend)
<i>b</i>	second polynomial in x (divisor)
<i>x</i>	a and b are polynomials in x
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

#### Returns

remainder of a(x) and b(x) in  $\mathbb{Q}[x]$

References [\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [dynallocate\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**7.1.3.571 decomp\_rational()**

```
ex GiNaC::decomp_rational (
    const ex & a,
    const ex & x)
```

Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .

**Parameters**

<i>a</i>	rational function in <i>x</i>
<i>x</i>	<i>a</i> is a function of <i>x</i>

**Returns**

decomposed function.

References [denom\(\)](#), [is\\_exactly\\_a\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [quo\(\)](#), and [rem\(\)](#).

**7.1.3.572 prem()**

```
ex GiNaC::prem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args)
```

Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

**Parameters**

<i>a</i>	first polynomial in <i>x</i> (dividend)
<i>b</i>	second polynomial in <i>x</i> (divisor)
<i>x</i>	<i>a</i> and <i>b</i> are polynomials in <i>x</i>
<i>check_args</i>	check whether <i>a</i> and <i>b</i> are polynomials with rational coefficients (defaults to "true")

**Returns**

pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [sr\\_gcd\(\)](#).

**7.1.3.573 sprem()**

```
ex GiNaC::sprem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args)
```

Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

**Parameters**

<i>a</i>	first polynomial in <i>x</i> (dividend)
----------	---

<i>b</i>	second polynomial in x (divisor)
<i>x</i>	a and b are polynomials in x
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

sparse pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

**7.1.3.574 divide()**

```
bool GiNaC::divide (
    const ex & a,
    const ex & b,
    ex & q,
    bool check_args)
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Q}[X]$ .

**Parameters**

<i>a</i>	first multivariate polynomial (dividend)
<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

"true" when exact division succeeds (quotient returned in q), "false" otherwise (q left untouched)

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [get\\_first\\_symbol\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [divide\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [gcd\(\)](#), [quo\(\)](#), and [rem\(\)](#).

**7.1.3.575 divide\_in\_z()**

```
bool GiNaC::divide_in_z (
    const ex & a,
    const ex & b,
    ex & q,
    sym_desc_vec::const_iterator var) [static]
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Z}[X]$ .

This functions works like [divide\(\)](#) but the input and output polynomials are in  $\mathbb{Z}[X]$  instead of  $\mathbb{Q}[X]$  (i.e. they have integer coefficients). Unlike [divide\(\)](#), it doesn't check whether the input polynomials really are integer polynomials, so be careful of what you pass in. Also, you have to run [get\\_symbol\\_stats\(\)](#) over the input polynomials before calling this function and pass an iterator to the first element of the [sym\\_desc](#) vector. This function is used internally by the [heur\\_gcd\(\)](#).

**Parameters**

<i>a</i>	first multivariate polynomial (dividend)
----------	--

<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs

**Returns**

"true" when exact division succeeds (the quotient is returned in q), "false" otherwise.

**See also**

[get\\_symbol\\_stats](#), [heur\\_gcd](#)

References [\\_ex0](#), [\\_ex1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [get\\_symbol\\_stats\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [divide\\_in\\_z\(\)](#), [heur\\_gcd\\_z\(\)](#), and [sr\\_gcd\(\)](#).

**7.1.3.576 sr\_gcd()**

```
ex GiNaC::sr_gcd (
    const ex & a,
    const ex & b,
    sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the subresultant PRS algorithm.

This function is used internally by [gcd\(\)](#).

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs

**Returns**

the GCD as a new expression

**See also**

[gcd](#)

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::content\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [pow\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), and [psi\(\)](#).

Referenced by [gcd\(\)](#).

### 7.1.3.577 interpolate()

```
ex GiNaC::interpolate (
    const ex & gamma,
    const numeric & xi,
    const ex & x,
    int degree_hint = 1) [static]
```

xi-adic polynomial interpolation

References [dynallocate\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), and [GiNaC::ex::smod\(\)](#).

Referenced by [heur\\_gcd\\_z\(\)](#).

### 7.1.3.578 heur\_gcd\_z()

```
bool GiNaC::heur_gcd_z (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

#### Parameters

<i>a</i>	first integer multivariate polynomial (expanded)
<i>b</i>	second integer multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs
<i>res</i>	the GCD (returned)

#### Returns

true if GCD was computed, false otherwise.

#### See also

[gcd](#)

#### Exceptions

<a href="#">gcdheu_failed()</a>	
---------------------------------	--

References [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::numeric::int\\_length\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [interpolate\(\)](#), [GiNaC::numeric::inverse\(\)](#), [iquo\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [isqrt\(\)](#), [GiNaC::ex::max\\_coefficient\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [heur\\_gcd\(\)](#), and [heur\\_gcd\\_z\(\)](#).



**7.1.3.579** `heur_gcd()`

```
bool GiNaC::heur_gcd (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

**Parameters**

<i>a</i>	first rational multivariate polynomial (expanded)
<i>b</i>	second rational multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs
<i>res</i>	the GCD (returned)

**Returns**

true if GCD was computed, false otherwise.

**See also**

[heur\\_gcd\\_z](#)  
[gcd](#)

References [heur\\_gcd\\_z\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), and [lcmcoeff\(\)](#).

Referenced by [gcd\(\)](#).

**7.1.3.580** `gcd_pf_pow()`

```
ex GiNaC::gcd_pf_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb) [static]
```

References [\\_ex1](#), [ex\\_to\(\)](#), [expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GINAC\\_ASSERT](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::ldegree\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_pow\(\)](#).

### 7.1.3.581 gcd\_pf\_mul()

```
ex GiNaC::gcd_pf_mul (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb) [static]
```

References [dynamallocate\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [GINAC\\_ASSERT](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_mul\(\)](#).

### 7.1.3.582 gcd() [1/2]

```
ex GiNaC::gcd (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    bool check_args,
    unsigned options)
```

Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Z}[X]$ .

Optionally also compute the cofactors of  $a$  and  $b$ , defined by  $a = ca * \text{gcd}(a, b)$  and  $b = cb * \text{gcd}(a, b)$ .

#### Parameters

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>ca</i>	pointer to expression that will receive the cofactor of $a$ , or nullptr
<i>cb</i>	pointer to expression that will receive the cofactor of $b$ , or nullptr
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")
<i>options</i>	see <a href="#">GiNaC::gcd_options</a>

#### Returns

the GCD as a new expression

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [get\\_symbol\\_stats\(\)](#), [heur\\_gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::gcd\\_options::no\\_heur\\_gcd](#), [GiNaC::gcd\\_options::no\\_part\\_fa](#), [GiNaC::subs\\_options::no\\_pattern](#), [pow\(\)](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [sr\\_gcd\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [GiNaC::gcd\\_options::use\\_sr\\_gcd](#).

Referenced by [GiNaC::ex::content\(\)](#), [dirichlet\\_character\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [GiNaC::add::normal\(\)](#), [sqrfree\\_yun\(\)](#), and [sr\\_gcd\(\)](#).

**7.1.3.583 gcd\_pf\_pow\_pow()**

```
ex GiNaC::gcd_pf_pow_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb) [static]
```

References [\\_ex1](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\\_pf\\_pow\(\)](#).

**7.1.3.584 lcm() [1/2]**

```
ex GiNaC::lcm (
    const ex & a,
    const ex & b,
    bool check_args)
```

Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

the LCM as a new expression

References [ex\\_to\(\)](#), [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_exactly\\_a\(\)](#), [lcm\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).

**7.1.3.585 sqrfree\_yun()**

```
epvector GiNaC::sqrfree_yun (
    const ex & a,
    const symbol & x) [static]
```

Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.

Used internally by [sqrfree\(\)](#).

**Parameters**

<i>a</i>	multivariate polynomial over $\mathbb{Z}[X]$ , treated here as univariate polynomial in $x$ (needs not be expanded).
<i>x</i>	variable to factor in

**Returns**

vector of expairs (factor, exponent), sorted by exponents

References [\\_ex1](#), [GiNaC::ex::diff\(\)](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [quo\(\)](#).

Referenced by [sqrfree\(\)](#), and [sqrfree\\_parfrac\(\)](#).

### 7.1.3.586 `sqrfree()`

```
ex GiNaC::sqrfree (
    const ex & a,
    const lst & l)
```

Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .

#### Parameters

<i>a</i>	multivariate polynomial over $\mathbb{Q}[X]$ (needs not be expanded)
<i>l</i>	lst of variables to factor in, may be left empty for autodetection

#### Returns

a square-free factorization of *a*.

#### Note

A polynomial  $p(X) \in C[X]$  is said *square-free* if, whenever any two polynomials  $q(X)$  and  $r(X)$  are such that

$$p(X) = q(X)^2 r(X),$$

we have  $q(X) \in C$ . This means that  $p(X)$  has no repeated factors, apart eventually from constants. Given a polynomial  $p(X) \in C[X]$ , we say that the decomposition

$$p(X) = b \cdot p_1(X)^{a_1} \cdot p_2(X)^{a_2} \cdots p_r(X)^{a_r}$$

is a *square-free factorization* of  $p(X)$  if the following conditions hold:

1.  $b \in C$  and  $b \neq 0$ ;
2.  $a_i$  is a positive integer for  $i = 1, \dots, r$ ;
3. the degree of the polynomial  $p_i$  is strictly positive for  $i = 1, \dots, r$ ;
4. the polynomial  $\prod_{i=1}^r p_i(X)$  is square-free.

Square-free factorizations need not be unique. For example, if  $a_i$  is even, we could change the polynomial  $p_i(X)$  into  $-p_i(X)$ . Observe also that the factors  $p_i(X)$  need not be irreducible polynomials.

References [\\_ex0](#), [GiNaC::container< C >::append\(\)](#), [ex\\_to\(\)](#), [get\\_symbol\\_stats\(\)](#), [is\\_a\(\)](#), [is\\_exactly\\_a\(\)](#), [lcm\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), [sqrfree\(\)](#), and [sqrfree\\_yun\(\)](#).

Referenced by [sqrfree\(\)](#).

### 7.1.3.587 `sqrfree_parfrac()`

```
ex GiNaC::sqrfree_parfrac (
    const ex & a,
    const symbol & x)
```

Compute square-free partial fraction decomposition of rational function  $a(x)$ .

#### Parameters

<i>a</i>	rational function over $\mathbb{Z}[x]$ , treated as univariate polynomial in <i>x</i>
----------	---

x	variable to factor in
---	-----------------------

**Returns**

decomposed rational function

References [\\_ex1](#), [GiNaC::basic::coeff\(\)](#), [coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [degree\(\)](#), [denom\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [quo\(\)](#), [rem\(\)](#), [rhs\(\)](#), [GiNaC::matrix::solve\(\)](#), [sqrtfree\\_yun\(\)](#), and [to\\_int\(\)](#).

**7.1.3.588 replace\_with\_symbol() [1/2]**

```
ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to repl, for a later application of [subs\(\)](#). An entry in the replacement table repl can be changed in some cases. If it was altered, we need to provide the modifier for the previously build expressions. The modifier is an (ordered) list, because those substitutions need to be done in the incremental order. As an example let us consider a rationalisation of the expression  $e = \exp(2x) \cdot \cos(\exp(2x)+1) \cdot \exp(x)$ . The first factor [GiNaC](#) denotes by something like symbol1 and will record:  $e = \text{symbol1} \cdot \cos(\text{symbol1} + 1) \cdot \exp(x)$   $\text{repl} = \{\text{symbol1} : \exp(2x)\}$  Similarly, the second factor would be denoted as symbol2 and we will have  $e = \text{symbol1} \cdot \text{symbol2} \cdot \exp(x)$   $\text{repl} = \{\text{symbol1} : \exp(2x), \text{symbol2} : \cos(\text{symbol1} + 1)\}$  Denoting the third term as symbol3 [GiNaC](#) is willing to re-think  $\exp(2x)$  as  $\text{symbol3}^2$  rather than just symbol1. Here are two issues: 1) The replacement "symbol1 -> symbol3^2" in the previous part of the expression needs to be done outside of the present routine; 2) The pair "symbol1 :  $\exp(2x)$ " shall be deleted from the replacement table repl. However, this will create illegal substitution "symbol2 :  $\cos(\text{symbol1} + 1)$ " with undefined symbol1. These both problems are mitigated through the additions of the record "symbol1==symbol3^2" to the list modifier. Changed length of the modifier signals to the calling code that the previous portion of the expression needs to be altered (it solves 1). Thus [GiNaC](#) can record now  $e = \text{symbol3}^2 \cdot \text{symbol2} \cdot \text{symbol3}$   $\text{repl} = \{\text{symbol2} : \cos(\text{symbol1} + 1), \text{symbol3} : \exp(x)\}$   $\text{modifier} = \{\text{symbol1} == \text{symbol3}^2\}$  Then, doing the backward substitutions the list modifier will be used to restore such iterative substitutions in the right way (this solves 2).

**See also**

[ex::normal](#)

References [\\_ex\\_1](#), [GiNaC::container< C >::append\(\)](#), [degree\(\)](#), [denom\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [exp\(\)](#), [GiNaC::ex::find\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [normal\(\)](#), [numer\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::basic::normal\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::basic::to\\_polynomial\(\)](#), [GiNaC::numeric::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::basic::to\\_rational\(\)](#), [GiNaC::numeric::to\\_rational\(\)](#), and [GiNaC::power::to\\_rational\(\)](#).

**7.1.3.589 replace\_with\_symbol()** [2/2]

```
ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to repl, and the symbol is returned.

See also

[basic::to\\_rational](#)  
[basic::to\\_polynomial](#)

References [dynallocate\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.590 frac\_cancel()**

```
ex GiNaC::frac_cancel (
    const ex & n,
    const ex & d) [static]
```

Fraction cancellation.

**Parameters**

<i>n</i>	numerator
<i>d</i>	denominator

**Returns**

cancelled fraction {n, d} as a list

References [\\_ex1](#), [\\_ex\\_1](#), [\\_num1\\_p](#), [GiNaC::numeric::denom\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [get\\_first\\_symbol\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [is\\_negative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::numeric::numer\(\)](#), and [GiNaC::ex::unit\(\)](#).

Referenced by [GiNaC::add::normal\(\)](#), and [GiNaC::mul::normal\(\)](#).

**7.1.3.591 find\_common\_factor()**

```
ex GiNaC::find_common_factor (
    const ex & e,
    ex & factor,
    exmap & repl) [static]
```

Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [dynallocate\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::ex::to\\_polynomial\(\)](#).

Referenced by [collect\\_common\\_factors\(\)](#), and [find\\_common\\_factor\(\)](#).

**7.1.3.592 collect\_common\_factors()**

```
ex GiNaC::collect_common_factors (
    const ex & e)
```

Collect common factors in sums.

This converts expressions like 'a\*(b\*x+b\*y)' to 'a\*b\*(x+y)'.

References [factor\(\)](#), [find\\_common\\_factor\(\)](#), [is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::power::to\\_polynomial\(\)](#).

**7.1.3.593 resultant()**

```
ex GiNaC::resultant (
    const ex & e1,
    const ex & e2,
    const ex & s)
```

Resultant of two expressions e1,e2 with respect to symbol s.

Method: Compute determinant of Sylvester matrix of e1,e2,s.

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::lddegree\(\)](#), and [GiNaC::info\\_flags::polynomial](#).

**7.1.3.594 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [27/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    numeric ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_csrc_cl_N > &::↵
do_print_csrc_cl_N. print_func< print_tree > &::do_print_tree. print_func< print_python_repr
> &::do_print_python_repr )
```

default ctor.

Numerically it initializes to an integer zero.

References [GiNaC::numeric::do\\_print\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::numeric::do\\_print\\_latex\(\)](#), [GiNaC::numeric::do\\_print\\_python\\_repr\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::value](#).

**7.1.3.595 make\_real\_float()**

```
const cln::cl_F GiNaC::make_real_float (
    const cln::cl_idcoded_float & dec) [static]
```

Construct a floating point number from sign, mantissa, and exponent.

Referenced by [read\\_real\\_float\(\)](#).

**7.1.3.596 read\_real\_float()**

```
const cln::cl_F GiNaC::read_real_float (
    std::istream & s) [static]
```

Read serialized floating point number.

References [make\\_real\\_float\(\)](#).

Referenced by [GiNaC::numeric::read\\_archive\(\)](#).

**7.1.3.597 GINAC\_BIND\_UNARCHIVER() [37/50]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    numeric )
```

**7.1.3.598 write\_real\_float()**

```
void GiNaC::write_real_float (
    std::ostream & s,
    const cln::cl_R & n) [static]
```

Referenced by [GiNaC::numeric::archive\(\)](#).

**7.1.3.599 print\_real\_number()**

```
void GiNaC::print_real_number (
    const print_context & c,
    const cln::cl_R & x) [static]
```

Helper function to print a real number in a nicer way than is CLN's default.

Instead of printing 42.0L0 this just prints 42.0 to ostream os and instead of 3.99168L7 it prints 3.99168E7. This is fine in [GiNaC](#) as long as it only uses cl\_LF and no other floating point types that we might want to visibly distinguish from cl\_LF.

See also

[numeric::print\(\)](#)

References [is\\_a\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::numeric::print\\_numeric\(\)](#), and [print\\_real\\_cl\\_N\(\)](#).



**7.1.3.600 print\_integer\_csrc()**

```
void GiNaC::print_integer_csrc (
    const print_context & c,
    const cln::cl_I & x) [static]
```

Helper function to print integer number in C++ source format.

See also

[numeric::print\(\)](#)

References [GiNaC::print\\_context::s](#).

Referenced by [print\\_real\\_csrc\(\)](#).

**7.1.3.601 print\_real\_csrc()**

```
void GiNaC::print_real_csrc (
    const print_context & c,
    const cln::cl_R & x) [static]
```

Helper function to print real number in C++ source format.

See also

[numeric::print\(\)](#)

References [denom\(\)](#), [numer\(\)](#), [print\\_integer\\_csrc\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\(\)](#).

**7.1.3.602 coerce()**

```
template<typename T1, typename T2>
bool GiNaC::coerce (
    T1 & dst,
    const T2 & arg) [inline], [static]
```

Referenced by [print\\_real\\_cl\\_N\(\)](#).

**7.1.3.603 coerce< int, cln::cl\_I >()**

```
template<>
bool GiNaC::coerce< int, cln::cl_I > (
    int & dst,
    const cln::cl_I & arg) [inline]
```

Check if CLN integer can be converted into int.

See also

<https://www.ginac.de/pipermail/cln-list/2006-October/000248.html>

**7.1.3.604 `coerce< unsigned int, cln::cl_I >()`**

```
template<>
bool GiNaC::coerce< unsigned int, cln::cl_I > (
    unsigned int & dst,
    const cln::cl_I & arg) [inline]
```

**7.1.3.605 `print_real_cl_N()`**

```
void GiNaC::print_real_cl_N (
    const print_context & c,
    const cln::cl_R & x) [static]
```

Helper function to print real number in C++ source format using `cl_N` types.

See also

[numeric::print\(\)](#)

References [coerce\(\)](#), [Digits](#), [print\\_real\\_number\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#).

**7.1.3.606 `exp()`**

```
const numeric GiNaC::exp (
    const numeric & x)
```

Exponential function.

Returns

arbitrary precision numerical  $\exp(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [Bernoulli\\_polynomial\(\)](#), [beta\\_evalf\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [csgn\\_power\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [exp\\_conjugate\(\)](#), [exp\\_deriv\(\)](#), [exp\\_eval\(\)](#), [exp\\_evalf\(\)](#), [exp\\_expand\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_power\(\)](#), [exp\\_real\\_part\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_val](#), [GiNaC::power::imag\\_part\(\)](#), [log\\_eval\(\)](#), [print\\_sym\\_pow\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::power::real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [tgamma\(\)](#).

### 7.1.3.607 log()

```
const numeric GiNaC::log (
    const numeric & x)
```

Natural logarithm.

#### Parameters

<i>x</i>	complex number
----------	----------------

#### Returns

arbitrary precision numerical  $\log(x)$ .

#### Exceptions

<i>pole_error("log()"</i>	logarithmic pole",0)
---------------------------	----------------------

References [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [exp\\_eval\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [log\\_conjugate\(\)](#), [log\\_eval\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [log\\_real\\_part\(\)](#), [log\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::power::real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [S\\_eval\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::Gt::tauToFundamental\(\)](#).

### 7.1.3.608 sin()

```
const numeric GiNaC::sin (
    const numeric & x)
```

Numeric sine (trigonometric function).

#### Returns

arbitrary precision numerical  $\sin(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_deriv\(\)](#), [cos\\_imag\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_conjugate\(\)](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tan\\_series\(\)](#).

### 7.1.3.609 cos()

```
const numeric GiNaC::cos (
    const numeric & x)
```

Numeric cosine (trigonometric function).

#### Returns

arbitrary precision numerical  $\cos(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_conjugate\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [GiNaC::power::real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_deriv\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), and [tan\\_series\(\)](#).

### 7.1.3.610 `tan()`

```
const numeric GiNaC::tan (  
    const numeric & x)
```

Numeric tangent (trigonometric function).

#### Returns

arbitrary precision numerical  $\tan(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), [tan\\_conjugate\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 7.1.3.611 `asin()`

```
const numeric GiNaC::asin (  
    const numeric & x)
```

Numeric inverse sine (trigonometric function).

#### Returns

arbitrary precision numerical  $\arcsin(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [asin\\_conjugate\(\)](#), [asin\\_eval\(\)](#), [asin\\_evalf\(\)](#), [cos\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

### 7.1.3.612 `acos()`

```
const numeric GiNaC::acos (  
    const numeric & x)
```

Numeric inverse cosine (trigonometric function).

#### Returns

arbitrary precision numerical  $\arccos(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [acos\\_conjugate\(\)](#), [acos\\_eval\(\)](#), [acos\\_evalf\(\)](#), [cos\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.3.613 atan()** [1/2]

```
const numeric GiNaC::atan (
    const numeric & x)
```

Numeric arcustangent.

**Parameters**

$x$	complex number
-----	----------------

**Returns**

atan( $x$ )

**Exceptions**

<i>pole_error("atan()"</i>	logarithmic pole",0) if $x=1$ or $x=-1$ .
----------------------------	---

References [\\_num1\\_p](#), [abs\(\)](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::numeric::real\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [atan2\\_eval\(\)](#), [atan2\\_evalf\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_eval\(\)](#), [atan\\_evalf\(\)](#), [atan\\_series\(\)](#), [cos\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.3.614 atan()** [2/2]

```
const numeric GiNaC::atan (
    const numeric & y,
    const numeric & x)
```

Numeric arcustangent of two arguments, analytically continued in a suitable way.

**Parameters**

$y$	complex number
$x$	complex number

**Returns**

$-i \cdot \log((x+iy)/\sqrt{x^2+y^2})$ , which is equal to  $\operatorname{atan}(y/x)$  if  $y$  and  $x$  are both real.

**Exceptions**

<i>pole_error("atan()"</i>	logarithmic pole",0) if $y/x=1$ or $y/x=-1$ .
----------------------------	---

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

### 7.1.3.615 `sinh()`

```
const numeric GiNaC::sinh (  
    const numeric & x)
```

Numeric hyperbolic sine (trigonometric function).

#### Returns

arbitrary precision numerical  $\sinh(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cosh\\_deriv\(\)](#), [cosh\\_imag\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_conjugate\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [sinh\\_real\\_part\(\)](#), and [tanh\\_series\(\)](#).

### 7.1.3.616 `cosh()`

```
const numeric GiNaC::cosh (  
    const numeric & x)
```

Numeric hyperbolic cosine (trigonometric function).

#### Returns

arbitrary precision numerical  $\cosh(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [cos\\_real\\_part\(\)](#), [cosh\\_conjugate\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [cosh\\_real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_deriv\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tanh\\_series\(\)](#).

### 7.1.3.617 `tanh()`

```
const numeric GiNaC::tanh (  
    const numeric & x)
```

Numeric hyperbolic tangent (trigonometric function).

#### Returns

arbitrary precision numerical  $\tanh(x)$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [REGISTER\\_FUNCTION\(\)](#), [tan\\_imag\\_part\(\)](#), [tanh\\_conjugate\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 7.1.3.618 asinh()

```
const numeric GiNaC::asinh (  
    const numeric & x)
```

Numeric inverse hyperbolic sine (trigonometric function).

#### Returns

arbitrary precision numerical asinh(x).

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [asinh\\_conjugate\(\)](#), [asinh\\_eval\(\)](#), [asinh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

### 7.1.3.619 acosh()

```
const numeric GiNaC::acosh (  
    const numeric & x)
```

Numeric inverse hyperbolic cosine (trigonometric function).

#### Returns

arbitrary precision numerical acosh(x).

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [acosh\\_conjugate\(\)](#), [acosh\\_eval\(\)](#), [acosh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

### 7.1.3.620 atanh()

```
const numeric GiNaC::atanh (  
    const numeric & x)
```

Numeric inverse hyperbolic tangent (trigonometric function).

#### Returns

arbitrary precision numerical atanh(x).

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [atanh\\_conjugate\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_evalf\(\)](#), [atanh\\_series\(\)](#), [cosh\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

**7.1.3.621 Li2\_series()** [2/2]

```
cln::cl_N GiNaC::Li2_series (
    const cln::cl_N & x,
    const cln::float_format_t & prec) [static]
```

Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.

**7.1.3.622 Li2\_projection()**

```
cln::cl_N GiNaC::Li2_projection (
    const cln::cl_N & x,
    const cln::float_format_t & prec) [static]
```

Folds Li2's argument inside a small rectangle to enhance convergence.

References [Li2\\_projection\(\)](#), and [Li2\\_series\(\)](#).

Referenced by [Li2\\_\(\)](#), and [Li2\\_projection\(\)](#).

**7.1.3.623 Li2\_()**

```
const cln::cl_N GiNaC::Li2_ (
    const cln::cl_N & value)
```

Numeric evaluation of Dilogarithm.

The domain is the entire complex plane, the branch cut lies along the positive real axis, starting at 1 and continuous with quadrant IV.

**Returns**

arbitrary precision numerical Li2(x).

References [Li2\\_projection\(\)](#).

Referenced by [Li2\(\)](#).

**7.1.3.624 Li2()**

```
const numeric GiNaC::Li2 (
    const numeric & x)
```

References [\\_num0\\_p](#), [Li2\\_\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [Li2\\_conjugate\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_evalf\(\)](#), [Li2\\_series\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).



**7.1.3.625 zeta()** [3/3]

```
const numeric GiNaC::zeta (
    const numeric & x)
```

Numeric evaluation of Riemann's Zeta function.

Currently works only for integer arguments.

References [GiNaC::numeric::is\\_real\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.626 guess\_precision()**

```
cln::float_format_t GiNaC::guess_precision (
    const cln::cl_N & x) [static]
```

Referenced by [lgamma\(\)](#), and [tgamma\(\)](#).

**7.1.3.627 lgamma()** [1/2]

```
const cln::cl_N GiNaC::lgamma (
    const cln::cl_N & x)
```

The Gamma function.

Use the Lanczos approximation. If the coefficients used here are not sufficiently many or sufficiently accurate, more can be calculated using the program `doc/examples/lanczos.cpp`. In that case, be sure to read the comments in that file.

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [lgamma\(\)](#), [log\(\)](#), [sin\(\)](#), and [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#).

Referenced by [beta\\_evalf\(\)](#), [lgamma\(\)](#), [lgamma\(\)](#), [lgamma\\_conjugate\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_evalf\(\)](#), [lgamma\\_series\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.628 lgamma()** [2/2]

```
const numeric GiNaC::lgamma (
    const numeric & x)
```

References [lgamma\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.629 tgamma()** [1/2]

```
const cln::cl_N GiNaC::tgamma (
    const cln::cl_N & x)
```

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [exp\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [sqrt\(\)](#), [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#), and [tgamma\(\)](#).

Referenced by [beta\\_eval\(\)](#), [beta\\_series\(\)](#), [psi2\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [tgamma\(\)](#), [tgamma\(\)](#), [tgamma\\_conjugate\(\)](#), [tgamma\\_deriv\(\)](#), [tgamma\\_eval\(\)](#), [tgamma\\_evalf\(\)](#), and [tgamma\\_series\(\)](#).

**7.1.3.630 tgamma()** [2/2]

```
const numeric GiNaC::tgamma (
    const numeric & x)
```

References [tgamma\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.631 psi()** [3/4]

```
const numeric GiNaC::psi (
    const numeric & x)
```

The psi function (aka polygamma function).

This is only a stub!

**7.1.3.632 psi()** [4/4]

```
const numeric GiNaC::psi (
    const numeric & n,
    const numeric & x)
```

The psi functions (aka polygamma functions).

This is only a stub!

**7.1.3.633 factorial()**

```
const numeric GiNaC::factorial (
    const numeric & n)
```

Factorial combinatorial function.

**Parameters**

$n$	integer argument $\geq 0$
-----	---------------------------

**Exceptions**

<i>range_error</i>	(argument must be integer $\geq 0$ )
--------------------	--------------------------------------

References [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [Bernoulli\\_polynomial\(\)](#), [GiNaC::Gt::eval\(\)](#), [factorial\\_conjugate\(\)](#), [factorial\\_eval\(\)](#), [factorial\\_evalf\(\)](#), [factorial\\_real\\_part\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [multinomial\\_coefficient\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [GiNaC::Gt::regularise\(\)](#), [S\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [symm\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), [tgamma\\_eval\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

**7.1.3.634 doublefactorial()**

```
const numeric GiNaC::doublefactorial (
    const numeric & n)
```

The double factorial combinatorial function.

(Scarcely used, but still useful in cases, like for exact results of  $\text{tgamma}(n+1/2)$  for instance.)

**Parameters**

$n$	integer argument $\geq -1$
-----	----------------------------

**Returns**

$n!! == n * (n-2) * (n-4) * \dots * (\{1|2\})$  with  $0!! == (-1)!! == 1$

**Exceptions**

<i>range_error</i>	(argument must be integer $\geq -1$ )
--------------------	---------------------------------------

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [tgamma\\_eval\(\)](#).

**7.1.3.635 binomial()**

```
const numeric GiNaC::binomial (
    const numeric & n,
    const numeric & k)
```

The Binomial coefficients.

It computes the binomial coefficients. For integer  $n$  and  $k$  and positive  $n$  this is the number of ways of choosing  $k$  objects from  $n$  distinct objects. If  $n$  is a negative integer, the formula  $\text{binomial}(n,k) == (-1)^k * \text{binomial}(k-n-1,k)$  (if  $k \geq 0$ )  $\text{binomial}(n,k) == (-1)^{(n-k)} * \text{binomial}(-k-1,n-k)$  (otherwise) is used to compute the result.

References [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [binomial\(\)](#), [GiNaC::numeric::compare\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [binomial\(\)](#), [binomial\\_conjugate\(\)](#), [binomial\\_eval\(\)](#), [binomial\\_evalf\(\)](#), [binomial\\_real\\_part\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::power::real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), and [GiNaC::Gt::tauToFundamental\(\)](#).

### 7.1.3.636 bernoulli()

```
const numeric GiNaC::bernoulli (
    const numeric & nn)
```

Bernoulli number.

The nth Bernoulli number is the coefficient of  $x^n/n!$  in the expansion of the function  $x/(e^x-1)$ .

#### Returns

the nth Bernoulli number (a rational number).

#### Exceptions

<i>range_error</i>	(argument must be integer $\geq 0$ )
--------------------	--------------------------------------

References [\\_num1\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [zeta1\\_eval\(\)](#).

### 7.1.3.637 fibonacci()

```
const numeric GiNaC::fibonacci (
    const numeric & n)
```

Fibonacci number.

The nth Fibonacci number  $F(n)$  is defined by the recurrence formula  $F(n)=F(n-1)+F(n-2)$  with  $F(0)=0$  and  $F(1)=1$ .

#### Parameters

<i>n</i>	an integer
----------	------------

#### Returns

the nth Fibonacci number  $F(n)$  (an integer number)

#### Exceptions

<i>range_error</i>	(argument must be an integer)
--------------------	-------------------------------

References [\\_num0\\_p](#), [fibonacci\(\)](#), [GiNaC::numeric::is\\_even\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [fibonacci\(\)](#).

**7.1.3.638 abs()**

```
const numeric GiNaC::abs (
    const numeric & x)
```

Absolute value.

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [abs\\_conjugate\(\)](#), [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_expand\(\)](#), [abs\\_expl\\_derivative\(\)](#), [abs\\_power\(\)](#), [abs\\_real\\_part\(\)](#), [adaptivesimpson\(\)](#), [atan\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [GiNaC::numeric::integer\\_content\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [log\\_real\\_part\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::numeric::max\\_coefficient\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::power::real\\_part\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [zeta1\\_eval\(\)](#).

**7.1.3.639 mod()**

```
const numeric GiNaC::mod (
    const numeric & a,
    const numeric & b)
```

Modulus (in positive representation).

In general,  $\text{mod}(a,b)$  has the sign of  $b$  or is zero, and  $\text{rem}(a,b)$  has the sign of  $a$  or is zero. This is different from Maple's  $\text{modp}$ , where the sign of  $b$  is ignored. It is in agreement with Mathematica's  $\text{Mod}$ .

**Returns**

$a \bmod b$  in the range  $[0, \text{abs}(b)-1]$  with sign of  $b$  if both are integer, 0 otherwise.

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [cos\\_eval\(\)](#), [exp\\_eval\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.3.640 smod()**

```
const numeric GiNaC::smod (
    const numeric & a_,
    const numeric & b_)
```

Modulus (in symmetric representation).

**Returns**

$a \bmod b$  in the range  $[-\text{iquo}(\text{abs}(b),2), \text{iquo}(\text{abs}(b),2)]$ .

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), and [GiNaC::numeric::smod\(\)](#).

**7.1.3.641 irem()** [1/2]

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b)
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b)` as far as sign conventions are concerned. In general, `mod(a,b)` has the sign of `b` or is zero, and `irem(a,b)` has the sign of `a` or is zero.

**Returns**

remainder of `a/b` if both are integer, 0 otherwise.

**Exceptions**

<code>overflow_error</code>	(division by zero) if <code>b</code> is zero.
-----------------------------	---

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), and [ifactor\(\)](#).

**7.1.3.642 irem()** [2/2]

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b,
    numeric & q)
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b,'q')` it obeys the relation `irem(a,b,q) == a - q*b`. In general, `mod(a,b)` has the sign of `b` or is zero, and `irem(a,b)` has the sign of `a` or is zero.

**Returns**

remainder of `a/b` and quotient stored in `q` if both are integer, 0 otherwise.

**Exceptions**

<code>overflow_error</code>	(division by zero) if <code>b</code> is zero.
-----------------------------	---

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.643 iquo()** [1/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b)
```

Numeric integer quotient.

Equivalent to Maple's iquo as far as sign conventions are concerned.

**Returns**

truncated quotient of a/b if both are integer, 0 otherwise.

**Exceptions**

<i>overflow_error</i>	(division by zero) if b is zero.
-----------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::power::eval\(\)](#), and [heur\\_gcd\\_z\(\)](#).

**7.1.3.644 iquo()** [2/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b,
    numeric & r)
```

Numeric integer quotient.

Equivalent to Maple's iquo(a,b,r) it obeys the relation  $r == a - \text{iquo}(a,b,r) * b$ .

**Returns**

truncated quotient of a/b and remainder stored in r if both are integer, 0 otherwise.

**Exceptions**

<i>overflow_error</i>	(division by zero) if b is zero.
-----------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.645 gcd()** [2/2]

```
const numeric GiNaC::gcd (
    const numeric & a,
    const numeric & b)
```

Greatest Common Divisor.

**Returns**

The GCD of two numbers if both are integer, a numerical 1 if they are not.

References [\\_num1\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.646 lcm()** [2/2]

```
const numeric GiNaC::lcm (
    const numeric & a,
    const numeric & b)
```

Least Common Multiple.

**Returns**

The LCM of two numbers if both are integer, the product of those two numbers if they are not.

References [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::mul\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**7.1.3.647 sqrt()** [1/2]

```
const numeric GiNaC::sqrt (
    const numeric & x)
```

Numeric square root.

If possible, sqrt(x) should respect squares of exact numbers, i.e. sqrt(4) should return integer 2.

**Parameters**

<i>x</i>	numeric argument
----------	------------------

**Returns**

square root of x. Branch cut along negative real axis, the negative real axis itself where  $\text{imag}(x)=0$  and  $\text{real}(x)<0$  belongs to the upper part where  $\text{imag}(x)>0$ .

References [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [clifford\\_norm\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticK\\_evalf\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [tgamma\(\)](#), and [tgamma\\_eval\(\)](#).

**7.1.3.648 isqrt()**

```
const numeric GiNaC::isqrt (
    const numeric & x)
```

Integer numeric square root.

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [heur\\_gcd\\_z\(\)](#).



**7.1.3.649 PiEvalf()**

```
ex GiNaC::PiEvalf ()
```

Floating point evaluation of Archimedes' constant Pi.

**7.1.3.650 EulerEvalf()**

```
ex GiNaC::EulerEvalf ()
```

Floating point evaluation of Euler's constant gamma.

**7.1.3.651 CatalanEvalf()**

```
ex GiNaC::CatalanEvalf ()
```

Floating point evaluation of Catalan's constant.

**7.1.3.652 operator<<() [6/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const _numeric_digits & e)
```

References [GiNaC::\\_numeric\\_digits::print\(\)](#).

**7.1.3.653 GINAC\_DECLARE\_UNARCHIVER() [39/52]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    numeric )
```

References [Digits](#), and [I](#).

**7.1.3.654 pow() [1/3]**

```
const numeric GiNaC::pow (
    const numeric & x,
    const numeric & y) [inline]
```

References [GiNaC::numeric::power\(\)](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [beta\\_eval\(\)](#), [clifford\\_inverse\(\)](#), [clifford\\_prime\(\)](#), [clifford\\_star\\_bar\(\)](#), [clifford\\_to\\_1st\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::Gt::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::evaluate\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::integrate\(\)](#), [interpolate\(\)](#), [kronecker\\_symbol\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [lcmcoeff\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_series\(\)](#), [log\\_series\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::op\(\)](#), [Order\\_power\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::Gt::regularise\(\)](#), [rem\(\)](#), [remove\\_dirac\\_ONE\(\)](#), [replace\\_with\\_symbol\(\)](#), [S\\_eval\(\)](#), [S\\_series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl](#), [sprem\(\)](#), [sqfree\\_parfrac\(\)](#), [sr\\_gcd\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::power::to\\_rational\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

**7.1.3.655 inverse()** [3/3]

```
const numeric GiNaC::inverse (
    const numeric & x) [inline]
```

References [GiNaC::numeric::inverse\(\)](#).

**7.1.3.656 step()**

```
numeric GiNaC::step (
    const numeric & x) [inline]
```

References [GiNaC::numeric::step\(\)](#).

Referenced by [abs\\_eval\(\)](#), [H\\_eval\(\)](#), [REGISTER\\_FUNCTION\(\)](#), [step\\_conjugate\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [step\\_real\\_part\(\)](#), and [step\\_series\(\)](#).

**7.1.3.657 csgn()**

```
int GiNaC::csgn (
    const numeric & x) [inline]
```

References [GiNaC::numeric::csgn\(\)](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [csgn\\_conjugate\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [csgn\\_power\(\)](#), [csgn\\_real\\_part\(\)](#), [csgn\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [log\\_series\(\)](#), and [REGISTER\\_FUNCTION\(\)](#).

**7.1.3.658 is\_zero()** [2/2]

```
bool GiNaC::is_zero (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_zero\(\)](#).

**7.1.3.659 is\_positive()**

```
bool GiNaC::is_positive (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_positive\(\)](#).

Referenced by [beta\\_eval\(\)](#).

**7.1.3.660 is\_negative()**

```
bool GiNaC::is_negative (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_negative\(\)](#).

Referenced by [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::power::eval\(\)](#), and [frac\\_cancel\(\)](#).

### 7.1.3.661 is\_integer()

```
bool GiNaC::is_integer (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_integer\(\)](#).

Referenced by [beta\\_eval\(\)](#), [binomial\\_eval\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::lddegree\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [GiNaC::power::series\(\)](#).

### 7.1.3.662 is\_pos\_integer()

```
bool GiNaC::is_pos_integer (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_pos\\_integer\(\)](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

### 7.1.3.663 is\_nonneg\_integer()

```
bool GiNaC::is_nonneg_integer (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#).

### 7.1.3.664 is\_even()

```
bool GiNaC::is_even (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_even\(\)](#).

Referenced by [abs\\_power\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 7.1.3.665 is\_odd()

```
bool GiNaC::is_odd (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_odd\(\)](#).

Referenced by [csgn\\_power\(\)](#).

### 7.1.3.666 is\_prime()

```
bool GiNaC::is_prime (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_prime\(\)](#).

### 7.1.3.667 is\_rational()

```
bool GiNaC::is_rational (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_rational\(\)](#).

Referenced by [beta\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [tgamma\\_eval\(\)](#).

### 7.1.3.668 is\_real()

```
bool GiNaC::is_real (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_real\(\)](#).

Referenced by [beta\\_eval\(\)](#), [fsolve\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

### 7.1.3.669 is\_cinteger()

```
bool GiNaC::is_cinteger (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_cinteger\(\)](#).

### 7.1.3.670 is\_crational()

```
bool GiNaC::is_crational (
    const numeric & x) [inline]
```

References [GiNaC::numeric::is\\_crational\(\)](#).

### 7.1.3.671 to\_int()

```
int GiNaC::to_int (
    const numeric & x) [inline]
```

References [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [color\\_trace\(\)](#), [dirac\\_trace\(\)](#), [H\\_evalf\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [S\\_eval\(\)](#), [GiNaC::power::series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

### 7.1.3.672 to\_long()

```
long GiNaC::to_long (
    const numeric & x) [inline]
```

References [GiNaC::numeric::to\\_long\(\)](#).

**7.1.3.673 to\_double()**

```
double GiNaC::to_double (
    const numeric & x) [inline]
```

References [GiNaC::numeric::to\\_double\(\)](#).

**7.1.3.674 real()**

```
const numeric GiNaC::real (
    const numeric & x) [inline]
```

References [GiNaC::numeric::real\(\)](#).

Referenced by [cosh\\_eval\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

**7.1.3.675 imag()**

```
const numeric GiNaC::imag (
    const numeric & x) [inline]
```

References [GiNaC::numeric::imag\(\)](#).

Referenced by [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [G2\\_eval\(\)](#), and [G2\\_evalf\(\)](#).

**7.1.3.676 numer() [2/2]**

```
const numeric GiNaC::numer (
    const numeric & x) [inline]
```

References [GiNaC::numeric::numer\(\)](#).

**7.1.3.677 denom() [2/2]**

```
const numeric GiNaC::denom (
    const numeric & x) [inline]
```

References [GiNaC::numeric::denom\(\)](#).

**7.1.3.678 exadd()**

```
const ex GiNaC::exadd (
    const ex & lh,
    const ex & rh) [inline], [static]
```

Used internally by [operator+\(\)](#) to add two ex objects.

References [dynallocate\(\)](#).

Referenced by [operator+\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator--\(\)](#), and [operator-=\(\)](#).

### 7.1.3.679 `exmul()`

```
const ex GiNaC::exmul (  
    const ex & lh,  
    const ex & rh) [inline], [static]
```

Used internally by [operator\\*\(\)](#) to multiply two `ex` objects.

References [GiNaC::return\\_types::commutative](#), [dynallocate\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [operator\\*\(\)](#), [operator\\*=\(\(\)\)](#), [operator/\(\)](#), and [operator/=\(\(\)\)](#).

### 7.1.3.680 `exminus()`

```
const ex GiNaC::exminus (  
    const ex & lh) [inline], [static]
```

Used internally by [operator-\(\)](#) and friends to change the sign of an argument.

References [\\_ex\\_1](#), and [dynallocate\(\)](#).

Referenced by [operator-\(\)](#), [operator-\(\)](#), and [operator-=\(\(\)\)](#).

### 7.1.3.681 `operator+()` [1/4]

```
const ex GiNaC::operator+ (  
    const ex & lh,  
    const ex & rh)
```

References [exadd\(\)](#).

### 7.1.3.682 `operator-()` [1/4]

```
const ex GiNaC::operator- (  
    const ex & lh,  
    const ex & rh)
```

References [exadd\(\)](#), and [exminus\(\)](#).

### 7.1.3.683 `operator*()` [1/2]

```
const ex GiNaC::operator* (  
    const ex & lh,  
    const ex & rh)
```

References [exmul\(\)](#).

**7.1.3.684 operator/()** [1/2]

```
const ex GiNaC::operator/ (
    const ex & lh,
    const ex & rh)
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**7.1.3.685 operator+()** [2/4]

```
const numeric GiNaC::operator+ (
    const numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::add\(\)](#).

**7.1.3.686 operator-()** [2/4]

```
const numeric GiNaC::operator- (
    const numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::sub\(\)](#).

**7.1.3.687 operator\*()** [2/2]

```
const numeric GiNaC::operator* (
    const numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::mul\(\)](#).

**7.1.3.688 operator/()** [2/2]

```
const numeric GiNaC::operator/ (
    const numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::div\(\)](#).

**7.1.3.689 operator+=()** [1/2]

```
ex & GiNaC::operator+= (
    ex & lh,
    const ex & rh)
```

References [exadd\(\)](#).

**7.1.3.690 operator-=()** [1/2]

```
ex & GiNaC::operator-= (
    ex & lh,
    const ex & rh)
```

References [exadd\(\)](#), and [exminus\(\)](#).

**7.1.3.691 operator\*=( )** [1/2]

```
ex & GiNaC::operator*= (
    ex & lh,
    const ex & rh)
```

References [exmul\(\)](#).

**7.1.3.692 operator/=( )** [1/2]

```
ex & GiNaC::operator/= (
    ex & lh,
    const ex & rh)
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**7.1.3.693 operator+=( )** [2/2]

```
numeric & GiNaC::operator+= (
    numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::add\(\)](#).

**7.1.3.694 operator-=( )** [2/2]

```
numeric & GiNaC::operator-= (
    numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::sub\(\)](#).

**7.1.3.695 operator\*=( )** [2/2]

```
numeric & GiNaC::operator*= (
    numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::mul\(\)](#).



**7.1.3.696 operator/=( ) [2/2]**

```
numeric & GiNaC::operator/= (
    numeric & lh,
    const numeric & rh)
```

References [GiNaC::numeric::div\(\)](#).

**7.1.3.697 operator+( ) [3/4]**

```
const ex GiNaC::operator+ (
    const ex & lh)
```

**7.1.3.698 operator-( ) [3/4]**

```
const ex GiNaC::operator- (
    const ex & lh)
```

References [exminus\(\)](#).

**7.1.3.699 operator+( ) [4/4]**

```
const numeric GiNaC::operator+ (
    const numeric & lh)
```

**7.1.3.700 operator-( ) [4/4]**

```
const numeric GiNaC::operator- (
    const numeric & lh)
```

References [\\_num\\_1\\_p](#).

**7.1.3.701 operator++( ) [1/4]**

```
ex & GiNaC::operator++ (
    ex & rh)
```

Expression prefix increment.

Adds 1 and returns incremented ex.

References [\\_ex1](#), and [exadd\(\)](#).

**7.1.3.702 operator--()** [1/4]

```
ex & GiNaC::operator-- (
    ex & rh)
```

Expression prefix decrement.

Subtracts 1 and returns decremented ex.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**7.1.3.703 operator++()** [2/4]

```
const ex GiNaC::operator++ (
    ex & lh,
    int )
```

Expression postfix increment.

Returns the ex and leaves the original incremented by 1.

References [\\_ex1](#), and [exadd\(\)](#).

**7.1.3.704 operator--()** [2/4]

```
const ex GiNaC::operator-- (
    ex & lh,
    int )
```

Expression postfix decrement.

Returns the ex and leaves the original decremented by 1.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**7.1.3.705 operator++()** [3/4]

```
numeric & GiNaC::operator++ (
    numeric & rh)
```

Numeric prefix increment.

Adds 1 and returns incremented number.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**7.1.3.706 operator--()** [3/4]

```
numeric & GiNaC::operator-- (
    numeric & rh)
```

Numeric prefix decrement.

Subtracts 1 and returns decremented number.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**7.1.3.707 operator++()** [4/4]

```
const numeric GiNaC::operator++ (
    numeric & lh,
    int )
```

Numeric postfix increment.

Returns the number and leaves the original incremented by 1.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**7.1.3.708 operator--()** [4/4]

```
const numeric GiNaC::operator-- (
    numeric & lh,
    int )
```

Numeric postfix decrement.

Returns the number and leaves the original decremented by 1.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**7.1.3.709 operator==( )**

```
const relational GiNaC::operator== (
    const ex & lh,
    const ex & rh)
```

References [GiNaC::relational::equal](#).

**7.1.3.710 operator!=( )**

```
const relational GiNaC::operator!= (
    const ex & lh,
    const ex & rh)
```

References [GiNaC::relational::not\\_equal](#).

#### 7.1.3.711 operator<()

```
const relational GiNaC::operator< (  
    const ex & lh,  
    const ex & rh)
```

References [GiNaC::relational::less](#).

#### 7.1.3.712 operator<=()

```
const relational GiNaC::operator<= (  
    const ex & lh,  
    const ex & rh)
```

References [GiNaC::relational::less\\_or\\_equal](#).

#### 7.1.3.713 operator>()

```
const relational GiNaC::operator> (  
    const ex & lh,  
    const ex & rh)
```

References [GiNaC::relational::greater](#).

#### 7.1.3.714 operator>=()

```
const relational GiNaC::operator>= (  
    const ex & lh,  
    const ex & rh)
```

References [GiNaC::relational::greater\\_or\\_equal](#).

#### 7.1.3.715 my\_ios\_index()

```
int GiNaC::my_ios_index () [static]
```

Referenced by [get\\_print\\_context\(\)](#), and [set\\_print\\_context\(\)](#).

#### 7.1.3.716 my\_ios\_callback()

```
void GiNaC::my_ios_callback (  
    std::ios_base::event ev,  
    std::ios_base & s,  
    int i) [static]
```

Referenced by [set\\_print\\_context\(\)](#).

**7.1.3.717 `get_print_context()`**

```
print_context * GiNaC::get_print_context (
    std::ios_base & s) [inline], [static]
```

References [my\\_ios\\_index\(\)](#).

Referenced by [get\\_print\\_options\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), and [set\\_print\\_options\(\)](#).

**7.1.3.718 `set_print_context()`**

```
void GiNaC::set_print_context (
    std::ios_base & s,
    const print_context & c) [static]
```

References [callback\\_registered](#), [my\\_ios\\_callback\(\)](#), [my\\_ios\\_index\(\)](#), and [GiNaC::print\\_context::options](#).

Referenced by [csrc\(\)](#), [csrc\\_cl\\_N\(\)](#), [csrc\\_double\(\)](#), [csrc\\_float\(\)](#), [dflt\(\)](#), [latex\(\)](#), [python\(\)](#), [python\\_repr\(\)](#), [set\\_print\\_options\(\)](#), and [tree\(\)](#).

**7.1.3.719 `get_print_options()`**

```
unsigned GiNaC::get_print_options (
    std::ios_base & s) [inline], [static]
```

References [get\\_print\\_context\(\)](#), and [GiNaC::print\\_context::options](#).

Referenced by [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

**7.1.3.720 `set_print_options()`**

```
void GiNaC::set_print_options (
    std::ostream & s,
    unsigned options) [static]
```

References [get\\_print\\_context\(\)](#), [GiNaC::print\\_context::options](#), and [set\\_print\\_context\(\)](#).

Referenced by [dflt\(\)](#), [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

**7.1.3.721 `operator<<()` [7/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const ex & e)
```

References [get\\_print\\_context\(\)](#), and [GiNaC::ex::print\(\)](#).

**7.1.3.722 operator>>() [3/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    ex & e)
```

**7.1.3.723 dflt()**

```
std::ostream & GiNaC::dflt (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#), and [set\\_print\\_options\(\)](#).

**7.1.3.724 latex()**

```
std::ostream & GiNaC::latex (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.725 python()**

```
std::ostream & GiNaC::python (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.726 python\_repr()**

```
std::ostream & GiNaC::python_repr (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.727 tree()**

```
std::ostream & GiNaC::tree (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#).

**7.1.3.728 csrc()**

```
std::ostream & GiNaC::csrc (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.729 csrc\_float()**

```
std::ostream & GiNaC::csrc_float (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.730 csrc\_double()**

```
std::ostream & GiNaC::csrc_double (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.731 csrc\_cl\_N()**

```
std::ostream & GiNaC::csrc_cl_N (
    std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**7.1.3.732 index\_dimensions()**

```
std::ostream & GiNaC::index_dimensions (
    std::ostream & os)
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**7.1.3.733 no\_index\_dimensions()**

```
std::ostream & GiNaC::no_index_dimensions (
    std::ostream & os)
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**7.1.3.734 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [28/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    power ,
    basic ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_python > &::do_↵
print_python. print_func< print_python_repr > &::do_print_python_repr. print_func< print_csrc_cl_N
> &::do_print_csrc_cl_N )
```

References [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_python\(\)](#), and [GiNaC::power::do\\_print\\_python\\_repr\(\)](#).

**7.1.3.735 print\_sym\_pow()**

```
void GiNaC::print_sym_pow (
    const print_context & c,
    const symbol & x,
    int exp) [static]
```

References [exp\(\)](#), [GiNaC::basic::print\(\)](#), [print\\_sym\\_pow\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::power::do\\_print\\_csrc\(\)](#), and [print\\_sym\\_pow\(\)](#).

**7.1.3.736 GINAC\_BIND\_UNARCHIVER() [38/50]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    power )
```

**7.1.3.737 GINAC\_DECLARE\_UNARCHIVER() [40/52]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    power )
```

**7.1.3.738 pow() [2/3]**

```
ex GiNaC::pow (
    const ex & b,
    const ex & e) [inline]
```

Symbolic exponentiation.

Returns a power-object as a new expression.

**Parameters**

<i>b</i>	the basis expression
<i>e</i>	the exponent expression

References [dynallocate\(\)](#).

**7.1.3.739 pow() [3/3]**

```
template<typename T1, typename T2>
ex GiNaC::pow (
    const T1 & b,
    const T2 & e) [inline]
```

References [dynallocate\(\)](#).



**7.1.3.740 sqrt()** [2/2]

```
ex GiNaC::sqrt (
    const ex & a) [inline]
```

Square root expression.

Returns a power-object with exponent 1/2.

References [\\_ex1\\_2](#).

**7.1.3.741 is\_a()** [3/3]

```
template<class T>
bool GiNaC::is_a (
    const print_context & obj) [inline]
```

Check if obj is a T, including base classes.

**7.1.3.742 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [29/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    pseries ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python > &::do_↵
print_python. print_func< print_python_repr > &::do_print_python_repr )
```

References [GiNaC::pseries::do\\_print\(\)](#), [GiNaC::pseries::do\\_print\\_latex\(\)](#), [GiNaC::pseries::do\\_print\\_python\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), and [GiNaC::pseries::do\\_print\\_tree\(\)](#).

**7.1.3.743 GINAC\_BIND\_UNARCHIVER()** [39/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    pseries )
```

**7.1.3.744 GINAC\_DECLARE\_UNARCHIVER()** [41/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    pseries )
```

**7.1.3.745 series\_to\_poly()**

```
ex GiNaC::series_to_poly (
    const ex & e) [inline]
```

Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.

The result is undefined if the expression does not contain a pseries object at its top level.

**Parameters**

<i>e</i>	expression
----------	------------

**Returns**

polynomial expression

**See also**

[is\\_a<>](#)

[pseries::convert\\_to\\_poly](#)

References [ex\\_to\(\)](#).

Referenced by [Bernoulli\\_polynomial\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), and [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#).

**7.1.3.746 is\_terminating()**

```
bool GiNaC::is_terminating (
    const pseries & s) [inline]
```

References [GiNaC::pseries::is\\_terminating\(\)](#).

**7.1.3.747 make\_return\_type\_t()**

```
template<typename T>
return_type_t GiNaC::make_return_type_t (
    const unsigned rl = 0) [inline]
```

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [GiNaC::add::return\\_type\\_tinfo\(\)](#), [GiNaC::clifford::return\\_type\\_tinfo\(\)](#), [GiNaC::color::return\\_type\\_tinfo\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::mul::return\\_type\\_tinfo\(\)](#), [GiNaC::ncmul::return\\_type\\_tinfo\(\)](#), and [GiNaC::function\\_options::set\\_return\\_type\(\)](#).

**7.1.3.748 set\_print\_func()** [1/2]

```
template<class Alg, class Ctx, class T, class C>
void GiNaC::set_print_func (
    void f(const T &, const C &c, unsigned) [extern]
```

Add or replace a print method.

**7.1.3.749 set\_print\_func()** [2/2]

```
template<class Alg, class Ctx, class T, class C>
void GiNaC::set_print_func (
    void(T::* f)(const C &, unsigned)) [extern]
```

Add or replace a print method.

**7.1.3.750 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [30/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    relational ,
    basic ,
    print_func< print_context > &::do_print, print_func< print_tree > &::do_print↔
    _tree, print_func< print_python_repr > &::do_print_python_repr )
```

References [GiNaC::relational::do\\_print\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), and [GiNaC::basic::do\\_print\\_tree\(\)](#).

**7.1.3.751 GINAC\_BIND\_UNARCHIVER()** [40/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    relational )
```

**7.1.3.752 print\_operator()**

```
void GiNaC::print_operator (
    const print_context & c,
    relational::operators o) [static]
```

References [GiNaC::relational::equal](#), [GiNaC::relational::greater](#), [GiNaC::relational::greater\\_or\\_equal](#), [GiNaC::relational::less](#), [GiNaC::relational::less\\_or\\_equal](#), [GiNaC::relational::not\\_equal](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::relational::do\\_print\(\)](#), and [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#).

**7.1.3.753 GINAC\_DECLARE\_UNARCHIVER()** [42/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    relational )
```

**7.1.3.754 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [31/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symbol ,
    basic ,
    print_func< print_context > &::do_print, print_func< print_latex > &::do↔
    print_latex, print_func< print_tree > &::do_print_tree, print_func< print_python_repr >
    &::do_print_python_repr )
```

References [GiNaC::symbol::do\\_print\(\)](#), [GiNaC::symbol::do\\_print\\_latex\(\)](#), [GiNaC::symbol::do\\_print\\_python\\_repr\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**7.1.3.755 `get_default_TeX_name()`**

```
const std::string & GiNaC::get_default_TeX_name (
    const std::string & name) [static]
```

Return default TeX name for symbol.

This recognizes some greek letters.

Referenced by [GiNaC::symbol::do\\_print\\_latex\(\)](#), and [GiNaC::symbol::get\\_TeX\\_name\(\)](#).

**7.1.3.756 `GINAC_BIND_UNARCHIVER()` [41/50]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symbol )
```

**7.1.3.757 `GINAC_BIND_UNARCHIVER()` [42/50]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    realsymbol )
```

**7.1.3.758 `GINAC_BIND_UNARCHIVER()` [43/50]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    possymbol )
```

References [GiNaC::symbol::next\\_serial](#).

**7.1.3.759 `GINAC_DECLARE_UNARCHIVER()` [43/52]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symbol )
```

**7.1.3.760 `GINAC_DECLARE_UNARCHIVER()` [44/52]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    realsymbol )
```

**7.1.3.761 `GINAC_DECLARE_UNARCHIVER()` [45/52]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    possymbol )
```

**7.1.3.762 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [32/34]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symmetry ,
    basic ,
    print_func< print_context > &::do_print, print_func< print_tree > &::do_print←
    _tree )
```

References [GiNaC::symmetry::do\\_print\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**7.1.3.763 GINAC\_BIND\_UNARCHIVER()** [44/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symmetry )
```

**7.1.3.764 index0()**

```
const symmetry & GiNaC::index0 () [static]
```

References [dynallocate\(\)](#), and [ex\\_to\(\)](#).

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**7.1.3.765 index1()**

```
const symmetry & GiNaC::index1 () [static]
```

References [dynallocate\(\)](#), and [ex\\_to\(\)](#).

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**7.1.3.766 index2()**

```
const symmetry & GiNaC::index2 () [static]
```

References [dynallocate\(\)](#), and [ex\\_to\(\)](#).

Referenced by [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**7.1.3.767 index3()**

```
const symmetry & GiNaC::index3 () [static]
```

References [dynallocate\(\)](#), and [ex\\_to\(\)](#).

Referenced by [antisymmetric4\(\)](#), and [symmetric4\(\)](#).

**7.1.3.768 not\_symmetric()**

```
const symmetry & GiNaC::not_symmetric ()
```

References [dynallocate\(\)](#), and [ex\\_to\(\)](#).

Referenced by [GiNaC::clifford::clifford\(\)](#), [GiNaC::clifford::clifford\(\)](#), [clifford\\_unit\(\)](#), [GiNaC::color::color\(\)](#), [GiNaC::color::color\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), and [GiNaC::indexed::read\\_archive\(\)](#).

**7.1.3.769 symmetric2()**

```
const symmetry & GiNaC::symmetric2 ()
```

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), [index1\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [clifford\\_unit\(\)](#), [delta\\_tensor\(\)](#), [dirac\\_gamma\(\)](#), [dirac\\_slash\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [lorentz\\_g\(\)](#), and [metric\\_tensor\(\)](#).

**7.1.3.770 symmetric3()**

```
const symmetry & GiNaC::symmetric3 ()
```

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [color\\_d\(\)](#).

**7.1.3.771 symmetric4()**

```
const symmetry & GiNaC::symmetric4 ()
```

References [dynallocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), [index3\(\)](#), and [GiNaC::symmetry::symmetric](#).

**7.1.3.772 antisymmetric2()**

```
const symmetry & GiNaC::antisymmetric2 ()
```

References [GiNaC::symmetry::antisymmetric](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), and [index1\(\)](#).

Referenced by [epsilon\\_tensor\(\)](#), and [spinor\\_metric\(\)](#).

**7.1.3.773 antisymmetric3()**

```
const symmetry & GiNaC::antisymmetric3 ()
```

References [GiNaC::symmetry::antisymmetric](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), [index1\(\)](#), and [index2\(\)](#).

Referenced by [color\\_f\(\)](#), and [epsilon\\_tensor\(\)](#).

**7.1.3.774 antisymmetric4()**

```
const symmetry & GiNaC::antisymmetric4 ()
```

References [GiNaC::symmetry::antisymmetric](#), [dynalocate\(\)](#), [ex\\_to\(\)](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [index3\(\)](#).

Referenced by [lorentz\\_eps\(\)](#).

**7.1.3.775 canonicalize()**

```
int GiNaC::canonicalize (
    exvector::iterator v,
    const symmetry & symm)
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

**Parameters**

<i>v</i>	Start of expression vector
<i><a href="#">symm</a></i>	Root node of symmetry tree

**Returns**

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

Referenced by [GiNaC::symmetry::canonicalize](#), [GiNaC::function::eval\(\)](#), and [GiNaC::indexed::eval\(\)](#).

**7.1.3.776 symm()**

```
ex GiNaC::symm (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last,
    bool asymmetric) [static]
```

References [GiNaC::container< C >::append\(\)](#), [class\\_info< OPT >::first](#), [dynalocate\(\)](#), [factorial\(\)](#), [GiNaC::subs\\_options::no\\_index\\_recurse](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [permutation\\_sign\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::symmetry::canonicalize](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::symmetry::sy\\_swap](#), [GiNaC::ex::symmetrize\(\)](#), and [symmetrize\(\)](#).

**7.1.3.777 symmetrize() [3/4]**

```
ex GiNaC::symmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last)
```

Symmetrize expression over a set of objects (symbols, indices).

References [class\\_info< OPT >::first](#), and [symm\(\)](#).

**7.1.3.778 antisymmetrize()** [3/4]

```
ex GiNaC::antisymmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last)
```

Antisymmetrize expression over a set of objects (symbols, indices).

References [class\\_info< OPT >::first](#), and [symm\(\)](#).

**7.1.3.779 symmetrize\_cyclic()** [3/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last)
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::container< C >::append\(\)](#), [class\\_info< OPT >::first](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), and [GiNaC::ex::subs\(\)](#).

**7.1.3.780 GINAC\_DECLARE\_UNARCHIVER()** [46/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symmetry )
```

**7.1.3.781 sy\_none()** [1/4]

```
symmetry GiNaC::sy_none () [inline]
```

**7.1.3.782 sy\_none()** [2/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2) [inline]
```

References [GiNaC::symmetry::none](#).

**7.1.3.783 sy\_none()** [3/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).



**7.1.3.784 sy\_none()** [4/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).

**7.1.3.785 sy\_symm()** [1/4]

```
symmetry GiNaC::sy_symm () [inline]
```

References [GiNaC::symmetry::set\\_type\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**7.1.3.786 sy\_symm()** [2/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2) [inline]
```

References [GiNaC::symmetry::symmetric](#).

**7.1.3.787 sy\_symm()** [3/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**7.1.3.788 sy\_symm()** [4/4]

```
symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**7.1.3.789 sy\_anti()** [1/4]

```
symmetry GiNaC::sy_anti () [inline]
```

References [GiNaC::symmetry::antisymmetric](#), and [GiNaC::symmetry::set\\_type\(\)](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**7.1.3.790 sy\_anti() [2/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2) [inline]
```

References [GiNaC::symmetry::antisymmetric](#).

**7.1.3.791 sy\_anti() [3/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**7.1.3.792 sy\_anti() [4/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**7.1.3.793 sy\_cycl() [1/4]**

```
symmetry GiNaC::sy_cycl () [inline]
```

References [GiNaC::symmetry::cyclic](#), and [GiNaC::symmetry::set\\_type\(\)](#).

**7.1.3.794 sy\_cycl() [2/4]**

```
symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2) [inline]
```

References [GiNaC::symmetry::cyclic](#).

**7.1.3.795 sy\_cycl() [3/4]**

```
symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**7.1.3.796 sy\_cycl()** [4/4]

```

symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**7.1.3.797 symmetrize()** [4/4]

```

ex GiNaC::symmetrize (
    const ex & e,
    const exvector & v) [inline]

```

Symmetrize expression over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**7.1.3.798 antisymmetrize()** [4/4]

```

ex GiNaC::antisymmetrize (
    const ex & e,
    const exvector & v) [inline]

```

Antisymmetrize expression over a set of objects (symbols, indices).

References [antisymmetrize\(\)](#), and [GiNaC::ex::begin\(\)](#).

**7.1.3.799 symmetrize\_cyclic()** [4/4]

```

ex GiNaC::symmetrize_cyclic (
    const ex & e,
    const exvector & v) [inline]

```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**7.1.3.800 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [33/34]

```

GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    tensdelta ,
    tensor ,
    print_func< print_dflt > &::do_print, print_func< print_latex > &::do_print_↵
    latex )

```

References [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**7.1.3.801 print\_func< print\_dflt >()** [3/3]

```
GiNaC::print_func< print_dflt > (
    &tensmetric::do_print ) &
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)

**7.1.3.802 GINAC\_BIND\_UNARCHIVER()** [45/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    minkmetric )
```

**7.1.3.803 GINAC\_BIND\_UNARCHIVER()** [46/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensepsilon )
```

**7.1.3.804 GINAC\_BIND\_UNARCHIVER()** [47/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensdelta )
```

**7.1.3.805 GINAC\_BIND\_UNARCHIVER()** [48/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensmetric )
```

**7.1.3.806 GINAC\_BIND\_UNARCHIVER()** [49/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinmetric )
```

**7.1.3.807 delta\_tensor()**

```
ex GiNaC::delta_tensor (
    const ex & i1,
    const ex & i2)
```

Create a delta tensor with specified indices.

The indices must be of class `idx` or a subclass. The delta tensor is always symmetric and its trace is the dimension of the index space.

**Parameters**

<i>i1</i>	First index
-----------	-------------

<i>i2</i>	Second index
-----------	--------------

**Returns**

newly constructed delta tensor

References [dynallocate\(\)](#), [is\\_a\(\)](#), and [symmetric2\(\)](#).

Referenced by [color\\_trace\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**7.1.3.808 metric\_tensor()**

```
ex GiNaC::metric_tensor (
    const ex & i1,
    const ex & i2)
```

Create a symmetric metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. A metric tensor with one covariant and one contravariant index is equivalent to the delta tensor.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed metric tensor

References [dynallocate\(\)](#), [is\\_a\(\)](#), and [symmetric2\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#).

**7.1.3.809 lorentz\_g()**

```
ex GiNaC::lorentz_g (
    const ex & i1,
    const ex & i2,
    bool pos_sig = false)
```

Create a Minkowski metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. The Lorentz metric is a symmetric tensor with a matrix representation of  $\text{diag}(1, -1, -1, \dots)$  (negative signature, the default) or  $\text{diag}(-1, 1, 1, \dots)$  (positive signature).

**Parameters**

<i>i1</i>	First index
-----------	-------------

<i>i2</i>	Second index
<i>pos_sig</i>	Whether the signature is positive

**Returns**

newly constructed Lorentz metric tensor

References [dynamlocate\(\)](#), [is\\_a\(\)](#), and [symmetric2\(\)](#).

Referenced by [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [dirac\\_trace\(\)](#), and [trace\\_string\(\)](#).

**7.1.3.810 spinor\_metric()**

```
ex GiNaC::spinor_metric (
    const ex & i1,
    const ex & i2)
```

Create a spinor metric tensor with specified indices.

The indices must be of class `spinidx` or a subclass and have a dimension of 2. The spinor metric is an antisymmetric tensor with a matrix representation of  $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ .

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed spinor metric tensor

References [antisymmetric2\(\)](#), [dynamlocate\(\)](#), [ex\\_to\(\)](#), and [is\\_a\(\)](#).

**7.1.3.811 epsilon\_tensor()** [1/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2)
```

Create an epsilon tensor in a Euclidean space with two indices.

The indices must be of class `idx` or a subclass, and have a dimension of 2.

**Parameters**

<i>i1</i>	First index
-----------	-------------

<i>i2</i>	Second index
-----------	--------------

**Returns**

newly constructed epsilon tensor

References [\\_ex2](#), [antisymmetric2\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**7.1.3.812 epsilon\_tensor()** [2/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2,
    const ex & i3)
```

Create an epsilon tensor in a Euclidean space with three indices.

The indices must be of class `idx` or a subclass, and have a dimension of 3.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

**Returns**

newly constructed epsilon tensor

References [\\_ex3](#), [antisymmetric3\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**7.1.3.813 lorentz\_eps()**

```
ex GiNaC::lorentz_eps (
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4,
    bool pos_sig = false)
```

Create an epsilon tensor in a Minkowski space with four indices.

The indices must be of class `varidx` or a subclass, and have a dimension of 4.

**Parameters**

<i>i1</i>	First index
-----------	-------------

<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index
<i>pos_sig</i>	Whether the signature of the metric is positive

**Returns**

newly constructed epsilon tensor

References [\\_ex4](#), [antisymmetric4\(\)](#), [dynallocate\(\)](#), [ex\\_to\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [dirac\\_trace\(\)](#).

**7.1.3.814 GINAC\_DECLARE\_UNARCHIVER()** [47/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensdelta )
```

**7.1.3.815 GINAC\_DECLARE\_UNARCHIVER()** [48/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensmetric )
```

**7.1.3.816 GINAC\_DECLARE\_UNARCHIVER()** [49/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    minkmetric )
```

**7.1.3.817 GINAC\_DECLARE\_UNARCHIVER()** [50/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinmetric )
```

**7.1.3.818 GINAC\_DECLARE\_UNARCHIVER()** [51/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensepsilon )
```

**7.1.3.819 log2()**

```
unsigned GiNaC::log2 (
    unsigned n)
```

Integer binary logarithm.

Referenced by [GiNaC::remember\\_table::remember\\_table\(\)](#).



**7.1.3.820 multinomial\_coefficient()**

```
const numeric GiNaC::multinomial_coefficient (
    const std::vector< unsigned > & p)
```

Compute the multinomial coefficient  $n!$

$/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.  $p$  is a partition of  $n$ .

References [GiNaC::numeric::add\(\)](#), [GiNaC::numeric::div\(\)](#), [factorial\(\)](#), and [GiNaC::numeric::mul\(\)](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

**7.1.3.821 rotate\_left()**

```
unsigned GiNaC::rotate_left (
    unsigned n) [inline]
```

Rotate bits of unsigned value by one bit to the left.

This can be necessary if the user wants to define its own hashes.

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::Gt::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), and [GiNaC::symmetry::calchash\(\)](#).

**7.1.3.822 compare\_pointers()**

```
template<class T>
int GiNaC::compare_pointers (
    const T * a,
    const T * b) [inline]
```

Compare two pointers (just to establish some sort of canonical order).

Returns

-1, 0, or 1

Referenced by [GiNaC::basic::compare\\_same\\_type\(\)](#).

**7.1.3.823 golden\_ratio\_hash()**

```
unsigned GiNaC::golden_ratio_hash (
    uintptr_t n) [inline]
```

Truncated multiplication with golden ratio, for computing hash values.

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), and [GiNaC::wildcard::calchash\(\)](#).

**7.1.3.824 permutation\_sign() [1/2]**

```
template<class It>
int GiNaC::permutation_sign (
    It first,
    It last)
```

References [class\\_info< OPT >::first](#), [swap\(\)](#), and [std::swap\(\)](#).

Referenced by [GiNaC::symmetry::canonicalize](#), [GiNaC::matrix::determinant\(\)](#), [dirac\\_trace\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), and [symm\(\)](#).

**7.1.3.825 permutation\_sign() [2/2]**

```
template<class It, class Cmp, class Swap>
int GiNaC::permutation_sign (
    It first,
    It last,
    Cmp comp,
    Swap swapit)
```

References [class\\_info< OPT >::first](#).

**7.1.3.826 shaker\_sort()**

```
template<class It, class Cmp, class Swap>
void GiNaC::shaker_sort (
    It first,
    It last,
    Cmp comp,
    Swap swapit)
```

References [class\\_info< OPT >::first](#).

Referenced by [GiNaC::symmetry::canonicalize](#), [find\\_free\\_and\\_dummy\(\)](#), and [rename\\_dummy\\_indices\(\)](#).

**7.1.3.827 cyclic\_permutation()**

```
template<class It, class Swap>
void GiNaC::cyclic_permutation (
    It first,
    It last,
    It new_first,
    Swap swapit)
```

References [class\\_info< OPT >::first](#).

Referenced by [GiNaC::symmetry::canonicalize](#).

**7.1.3.828 format\_index\_value()** [1/2]

```
template<typename T>
std::enable_if< has_distance< T >::value, typename std::iterator_traits< T >::difference_type
>::type GiNaC::format_index_value (
    const T & a,
    const T & b)
```

For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.

However, we may print the difference to the starting point.

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

**7.1.3.829 format\_index\_value()** [2/2]

```
template<typename T>
std::enable_if<!has_distance< T >::value, T >::type GiNaC::format_index_value (
    const T & a,
    const T & b)
```

For all other cases we simply print the value.

**7.1.3.830 operator<<()** [8/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const basic_multi_iterator< T > & v) [inline]
```

Output operator.

A multi\_iterator prints out as [basic\\_multi\\_iterator](#)(  $n_0, n_1, \dots$ ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.831 operator<<()** [9/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered< T > & v) [inline]
```

Output operator.

A multi\_iterator\_ordered prints out as [multi\\_iterator\\_ordered](#)(  $n_0, n_1, \dots$ ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.832 operator<<()** [10/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< T > & v) [inline]
```

Output operator.

A `multi_iterator_ordered_eq` prints out as `multi_iterator_ordered_eq(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.833 operator<<()** [11/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< T > & v) [inline]
```

Output operator.

A `multi_iterator_ordered_eq_indv` prints out as `multi_iterator_ordered_eq_indv(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.834 operator<<()** [12/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter< T > & v) [inline]
```

Output operator.

A `multi_iterator_counter` prints out as `multi_iterator_counter(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.835 operator<<()** [13/16]

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< T > & v) [inline]
```

Output operator.

A `multi_iterator_counter_indv` prints out as `multi_iterator_counter_indv(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**7.1.3.836 operator<<() [14/16]**

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_permutation< T > & v) [inline]
```

Output operator.

A `multi_iterator_permutation` prints out as `multi_iterator_permutation(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**7.1.3.837 operator<<() [15/16]**

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< T > & v) [inline]
```

Output operator.

A `multi_iterator_shuffle` prints out as `multi_iterator_shuffle(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**7.1.3.838 operator<<() [16/16]**

```
template<class T>
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< T > & v) [inline]
```

Output operator.

A `multi_iterator_shuffle_prime` prints out as `multi_iterator_shuffle_prime(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**7.1.3.839 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [34/34]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    wildcard ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
    _tree. print_func< print_python_repr > &::do_print_python_repr )
```

References `GiNaC::wildcard::do_print()`, `GiNaC::wildcard::do_print_python_repr()`, `GiNaC::wildcard::do_print_tree()`, `GiNaC::status_flags::evaluated`, and `GiNaC::status_flags::expanded`.

**7.1.3.840 GINAC\_BIND\_UNARCHIVER()** [50/50]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    wildcard )
```

**7.1.3.841 haswild()**

```
bool GiNaC::haswild (
    const ex & x)
```

Check whether x has a wildcard anywhere as a subexpression.

References [haswild\(\)](#), [is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::integral::eval\(\)](#), and [haswild\(\)](#).

**7.1.3.842 GINAC\_DECLARE\_UNARCHIVER()** [52/52]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    wildcard )
```

**7.1.3.843 wild()**

```
ex GiNaC::wild (
    unsigned label = 0) [inline]
```

Create a wildcard object with the specified label.

Referenced by [GiNaC::spmapkey::spmapkey\(\)](#).

**7.1.4 Variable Documentation****7.1.4.1 unarch\_table\_instance**

```
unarchive_table_t GiNaC::unarch_table_instance [static]
```

**7.1.4.2 map\_evalm**

```
GiNaC::evalm_map_function GiNaC::map_evalm
```

Referenced by [GiNaC::basic::evalm\(\)](#).

**7.1.4.3 map\_eval\_integ**

```
GiNaC::eval_integ_map_function GiNaC::map_eval_integ
```

Referenced by [GiNaC::basic::eval\\_integ\(\)](#).

7.1.4.4 `class_info< OPT >::first`

```
template<class OPT>
class_info<OPT>* GiNaC::class_info< OPT >::first = nullptr
```

Referenced by [antisymmetrize\(\)](#), [GiNaC::symmetry::canonicalize](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [cyclic\\_permutation\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::integrate\(\)](#), [permutation\\_sign\(\)](#), [permutation\\_sign\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [shaker\\_sort\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [symm\(\)](#), [symmetrize\(\)](#), [symmetrize\\_cyclic\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

7.1.4.5 `class_info< OPT >::parents_identified`

```
template<class OPT>
bool GiNaC::class_info< OPT >::parents_identified = false
```

7.1.4.6 `tensor`

```
GiNaC::tensor
```

7.1.4.7 `Pi`

```
const constant GiNaC::Pi (
    "Pi" ,
    PiEvalf ,
    "\\pi" ,
    domain::positive )
```

Pi.

(3.14159...) Diverts straight into CLN for [evalf\(\)](#).

Referenced by [acos\\_eval\(\)](#), [acosh\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_eval\(\)](#), [EllipticK\\_evalf\(\)](#), [EllipticK\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [exp\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [H\\_evalf\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [log\\_eval\(\)](#), [log\\_series\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::constant::read\\_archive\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tan\\_series\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_series\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), [tgamma\\_eval\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

7.1.4.8 `Euler`

```
const constant GiNaC::Euler (
    "Euler" ,
    EulerEvalf ,
    "\\gamma_E" ,
    domain::positive )
```

Euler's constant.

(0.57721...) Sometimes called Euler-Mascheroni constant. Diverts straight into CLN for [evalf\(\)](#).

Referenced by [psi1\\_eval\(\)](#), and [GiNaC::constant::read\\_archive\(\)](#).

#### 7.1.4.9 Catalan

```
const constant GiNaC::Catalan (
    "Catalan" ,
    CatalanEvalf ,
    "G" ,
    domain::positive )
```

Catalan's constant.

(0.91597...) Diverts straight into CLN for [evalf\(\)](#).

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), and [GiNaC::constant::read\\_archive\(\)](#).

#### 7.1.4.10 crctab

```
unsigned const GiNaC::crctab[256] [static]
```

Referenced by [crc32\(\)](#).

#### 7.1.4.11 library\_initializer

```
library\_init GiNaC::library_initializer [static]
```

For construction of flyweights, etc.

#### 7.1.4.12 \_num0\_bp

```
const basic * GiNaC::_num0_bp
```

Referenced by [GiNaC::ex::ex\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.13 idx

```
GiNaC::idx
```

Referenced by [expand\\_dummy\\_sum\(\)](#).

#### 7.1.4.14 force\_include\_tgamma

```
unsigned GiNaC::force_include_tgamma = tgamma\_SERIAL::serial
```

#### 7.1.4.15 force\_include\_zeta1

```
unsigned GiNaC::force_include_zeta1 = zeta1\_SERIAL::serial
```



#### 7.1.4.16 GINAC\_BIND\_UNARCHIVER

```
template<>
GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(lst, basic, print_func< print_context >(&lst::do_print).
print_func< print_tree >(&lst::do_print_tree)) template<> bool ls GiNaC::GINAC_BIND_UNARCHIVER(
  (lst) (
    lst )
```

Specialization of `container::info()` for `lst`.

Referenced by `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()`.

#### 7.1.4.17 I

```
const numeric GiNaC::I = numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))
```

Imaginary unit.

This is not a constant but a numeric since we are natively handing complex numbers anyways, so in each expression containing an `I` it is automatically eval'ed away anyhow.

Referenced by `acosh_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `color_h()`, `GiNaC::su3f::contract_with()`, `cosh_eval()`, `csgn_eval()`, `dirac_trace()`, `eta_eval()`, `eta_evalf()`, `eta_imag_part()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `GINAC_DECLARE_UNARCHIVER()`, `H_evalf()`, `GiNaC::numeric::has()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::numeric::normal()`, `GiNaC::Gt::qExpand()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `sinh_eval()`, `step_eval()`, `tanh_eval()`, `tanh_series()`, `GiNaC::Gt::tauToFundamental()`, `GiNaC::numeric::to_polynomial()`, `GiNaC::numeric::to_rational()`, and `GiNaC::Gt::zisToFundamental()`.

#### 7.1.4.18 Digits

```
_numeric_digits GiNaC::Digits
```

Accuracy in decimal digits.

Only object of this type! Can be set using assignment from C++ unsigned ints and evaluated like any built-in type.

Referenced by `GiNaC::integration_kernel::get_numerical_value_impl()`, `GINAC_DECLARE_UNARCHIVER()`, `iterated_integral_evalf_impl()`, `GiNaC::numeric::numeric()`, `print_real_cl_N()`, `GiNaC::Gt::qExpand()`, and `zeta1_evalf()`.

#### 7.1.4.19 next\_print\_context\_id

```
unsigned GiNaC::next_print_context_id = 0
```

Next free ID for `print_context` types.

#### 7.1.4.20 `structure< T, CP >::reg_info`

```
template<class T, template< class > class CP>
registered_class_info GiNaC::structure< T, CP >::reg_info = registered_class_info(registered_class_options(st
"basic", typeid(structure<T, CP>)))
```

Referenced by [GiNaC::basic::print\\_dispatch\(\)](#).

#### 7.1.4.21 `version_major`

```
const int GiNaC::version_major = GINACLIB_MAJOR_VERSION
```

#### 7.1.4.22 `version_minor`

```
const int GiNaC::version_minor = GINACLIB_MINOR_VERSION
```

#### 7.1.4.23 `version_micro`

```
const int GiNaC::version_micro = GINACLIB_MICRO_VERSION
```

#### 7.1.4.24 `_num_120_p`

```
const numeric * GiNaC::_num_120_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.25 `_ex_120`

```
const ex GiNaC::_ex_120 = ex(*_num_120_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.26 `_num_60_p`

```
const numeric * GiNaC::_num_60_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.27 `_ex_60`

```
const ex GiNaC::_ex_60 = ex(*_num_60_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.28 `_num_48_p`

```
const numeric * GiNaC::_num_48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.29 `_ex_48`

```
const ex GiNaC::_ex_48 = ex(*_num_48_p)
```

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.30 `_num_30_p`

```
const numeric * GiNaC::_num_30_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.31 `_ex_30`

```
const ex GiNaC::_ex_30 = ex(*_num_30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.32 `_num_25_p`

```
const numeric * GiNaC::_num_25_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.33 `_ex_25`

```
const ex GiNaC::_ex_25 = ex(*_num_25_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.34 `_num_24_p`

```
const numeric * GiNaC::_num_24_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.35 `_ex_24`

```
const ex GiNaC::_ex_24 = ex(*_num_24_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.36 `_num_20_p`

```
const numeric * GiNaC::_num_20_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.37 `_ex_20`

```
const ex GiNaC::_ex_20 = ex(*_num_20_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.38 `_num_18_p`

```
const numeric * GiNaC::_num_18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.39 `_ex_18`

```
const ex GiNaC::_ex_18 = ex(*_num_18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.40 `_num_15_p`

```
const numeric * GiNaC::_num_15_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.41 `_ex_15`

```
const ex GiNaC::_ex_15 = ex(*_num_15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.42 `_num_12_p`

```
const numeric * GiNaC::_num_12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.43 `_ex_12`

```
const ex GiNaC::_ex_12 = ex(*_num_12_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.44 `_num_11_p`

```
const numeric * GiNaC::_num_11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.45 `_ex_11`

```
const ex GiNaC::_ex_11 = ex(*_num_11_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.46 `_num_10_p`

```
const numeric * GiNaC::_num_10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.47 `_ex_10`

```
const ex GiNaC::_ex_10 = ex(*_num_10_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.48 `_num_9_p`

```
const numeric * GiNaC::_num_9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.49 `_ex_9`

```
const ex GiNaC::_ex_9 = ex(*_num_9_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.50 `_num_8_p`

```
const numeric * GiNaC::_num_8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.51 `_ex_8`

```
const ex GiNaC::_ex_8 = ex(*_num_8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.52 `_num_7_p`

```
const numeric * GiNaC::_num_7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.53 `_ex_7`

```
const ex GiNaC::_ex_7 = ex(*_num_7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.54 `_num_6_p`

```
const numeric * GiNaC::_num_6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.55 `_ex_6`

```
const ex GiNaC::_ex_6 = ex(*_num_6_p)
```

Referenced by [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.56 `_num_5_p`

```
const numeric * GiNaC::_num_5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.57 `_ex_5`

```
const ex GiNaC::_ex_5 = ex(*_num_5_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.58 `_num_4_p`

```
const numeric * GiNaC::_num_4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.59 `_ex_4`

```
const ex GiNaC::_ex_4 = ex(*_num_4_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.60** `_num_3_p`

```
const numeric * GiNaC::_num_3_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.61** `_ex_3`

```
const ex GiNaC::_ex_3 = ex(*_num_3_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.62** `_num_2_p`

```
const numeric * GiNaC::_num_2_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [tgamma\\_eval\(\)](#).

**7.1.4.63** `_ex_2`

```
const ex GiNaC::_ex_2 = ex(*_num_2_p)
```

Referenced by [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.64** `_num_1_p`

```
const numeric * GiNaC::_num_1_p
```

Referenced by [acos\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [beta\\_eval\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi1\\_eval\(\)](#), and [psi2\\_eval\(\)](#).

**7.1.4.65** `_ex_1`

```
const ex GiNaC::_ex_1 = ex(*_num_1_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [GiNaC::diracgamma5::conjugate\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [exminus\(\)](#), [exp\\_eval\(\)](#), [exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [frac\\_cancel\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_deriv\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [operator--\(\)](#), [operator-\(\)](#), [operator/\(\)](#), [operator/=\(\(\)\)](#), [psi1\\_series\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [replace\\_with\\_symbol\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.66** `_num_1_2_p`

```
const numeric * GiNaC::_num_1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.67** `_ex_1_2`

```
const ex GiNaC::_ex_1_2 = ex(*_num_1_2_p)
```

Referenced by [acos\\_deriv\(\)](#), [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [asin\\_deriv\(\)](#), [asin\\_eval\(\)](#), [asinh\\_deriv\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.68** `_num_1_3_p`

```
const numeric * GiNaC::_num_1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.69** `_ex_1_3`

```
const ex GiNaC::_ex_1_3 = ex(*_num_1_3_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.70** `_num_1_4_p`

```
const numeric * GiNaC::_num_1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.71** `_ex_1_4`

```
const ex GiNaC::_ex_1_4 = ex(*_num_1_4_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.72** `_num0_p`

```
const numeric * GiNaC::_num0_p
```

Referenced by [GiNaC::numeric::add\\_dyn\(\)](#), [atan\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [fibonacci\(\)](#), [GiNaC::numeric::has\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [iquo\(\)](#), [iquo\(\)](#), [irem\(\)](#), [irem\(\)](#), [isqrt\(\)](#), [Li2\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [mod\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [sin\\_eval\(\)](#), [smod\(\)](#), [GiNaC::numeric::sub\\_dyn\(\)](#), and [tan\\_eval\(\)](#).



**7.1.4.73 `_ex0`**

```
const ex GiNaC::_ex0 = ex(*_num0_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_eval\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [asinh\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [base\\_and\\_index\(\)](#), [beta\\_eval\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::relational::canonical\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_trace\(\)](#), [GiNaC::ex::content\(\)](#), [cos\\_eval\(\)](#), [csgn\\_series\(\)](#), [GiNaC::expairseq::default\\_overall\\_coeff\(\)](#), [GiNaC::basic::derivative\(\)](#), [GiNaC::constant::derivative\(\)](#), [GiNaC::idx::derivative\(\)](#), [GiNaC::indexed::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [dirac\\_trace\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_eval\(\)](#), [EllipticK\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_valf\(\)](#), [eta\\_series\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::indexed::expand\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [G2\\_eval\(\)](#), [G2\\_valf\(\)](#), [G3\\_eval\(\)](#), [G3\\_valf\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [gcd\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\\_OPT\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_deriv\(\)](#), [Li\\_eval\(\)](#), [Li\\_valf\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [Order\\_eval\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [rem\(\)](#), [S\\_deriv\(\)](#), [S\\_eval\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::indexed::simplify\\_indexed\\_product\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [sprem\(\)](#), [sqrfree\(\)](#), [sr\\_gcd\(\)](#), [step\\_series\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [zeta1\\_deriv\(\)](#), [zeta1\\_eval\(\)](#), [zeta2\\_deriv\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.74 `_num1_4_p`**

```
const numeric * GiNaC::_num1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.75 `_ex1_4`**

```
const ex GiNaC::_ex1_4 = ex(*_num1_4_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.76 `_num1_3_p`**

```
const numeric * GiNaC::_num1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.77 `_ex1_3`**

```
const ex GiNaC::_ex1_3 = ex(*_num1_3_p)
```

Referenced by [acos\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.78 `_num1_2_p`

```
const numeric * GiNaC::_num1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), [psi2\\_eval\(\)](#), and [tgamma\\_eval\(\)](#).

#### 7.1.4.79 `_ex1_2`

```
const ex GiNaC::_ex1_2 = ex(*_num1_2_p)
```

Referenced by [acos\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [sin\\_eval\(\)](#), [sqrt\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.80 `_num1_p`

```
const numeric * GiNaC::_num1_p
```

Referenced by [acos\\_conjugate\(\)](#), [acosh\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [bernoulli\(\)](#), [binomial\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\\_dyn\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::basic::integer\\_content\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), [Li2\\_conjugate\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [GiNaC::basic::max\\_coefficient\(\)](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [multiply\\_lcm\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

#### 7.1.4.81 `_ex1`

```
const ex GiNaC::_ex1 = ex(*_num1_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [asin\\_eval\(\)](#), [asinh\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [base\\_and\\_index\(\)](#), [beta\\_eval\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_trace\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::cliffordunit::contract\\_with\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [GiNaC::mul::default\\_overall\\_coeff\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [dirac\\_trace\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::mul::evalm\(\)](#), [exp\\_eval\(\)](#), [GiNaC::expairseq::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [GiNaC::ncmul::hold\\_ncmul\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::matrix::inverse\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [Li\\_series\(\)](#),

[GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::symbol::normal\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [Order\\_eval\(\)](#), [Order\\_series\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [psi1\\_deriv\(\)](#), [psi1\\_series\(\)](#), [psi2\\_deriv\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [S\\_series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::indexed::simplify\\_indexed\\_product](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::expairseq::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [sprem\(\)](#), [sqrtfree\\_parfrac\(\)](#), [sqrtfree\\_yun\(\)](#), [sr\\_gcd\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tgamma\\_series\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::ex::unit\(\)](#), [unit\\_matrix\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [zeta1\\_deriv\(\)](#), [zeta2\\_deriv\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.82 `_num2_p`

```
const numeric * GiNaC::_num2_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [Li2\\_series\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [GiNaC::matrix::pow\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

#### 7.1.4.83 `_ex2`

```
const ex GiNaC::_ex2 = ex(*_num2_p)
```

Referenced by [acos\\_deriv\(\)](#), [asin\\_deriv\(\)](#), [asinh\\_deriv\(\)](#), [atan2\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atanh\\_deriv\(\)](#), [canonicalize\\_clifford\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [csgn\\_power\(\)](#), [epsilon\\_tensor\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [product\\_to\\_exvector\(\)](#), [psi1\\_eval\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.84 `_num3_p`

```
const numeric * GiNaC::_num3_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.85 `_ex3`

```
const ex GiNaC::_ex3 = ex(*_num3_p)
```

Referenced by [color\\_trace\(\)](#), [cos\\_eval\(\)](#), [epsilon\\_tensor\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.86 `_num4_p`

```
const numeric * GiNaC::_num4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.87 `_ex4`

```
const ex GiNaC::_ex4 = ex(*_num4_p)
```

Referenced by [dirac\\_trace\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [lorentz\\_eps\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.88 `_num5_p`

```
const numeric * GiNaC::_num5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 7.1.4.89 `_ex5`

```
const ex GiNaC::_ex5 = ex(*_num5_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.90 `_num6_p`

```
const numeric * GiNaC::_num6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 7.1.4.91 `_ex6`

```
const ex GiNaC::_ex6 = ex(*_num6_p)
```

Referenced by [cos\\_eval\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.92 `_num7_p`

```
const numeric * GiNaC::_num7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.93 `_ex7`

```
const ex GiNaC::_ex7 = ex(*_num7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.94 `_num8_p`

```
const numeric * GiNaC::_num8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.95 `_ex8`

```
const ex GiNaC::_ex8 = ex(*_num8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.96 `_num9_p`

```
const numeric * GiNaC::_num9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.97 `_ex9`

```
const ex GiNaC::_ex9 = ex(*_num9_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.98 `_num10_p`

```
const numeric * GiNaC::_num10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 7.1.4.99 `_ex10`

```
const ex GiNaC::_ex10 = ex(*_num10_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.100 `_num11_p`

```
const numeric * GiNaC::_num11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.101 `_ex11`

```
const ex GiNaC::_ex11 = ex(*_num11_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.102 `_num12_p`

```
const numeric * GiNaC::_num12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.103 `_ex12`

```
const ex GiNaC::_ex12 = ex(*_num12_p)
```

Referenced by [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.104 `_num15_p`

```
const numeric * GiNaC::_num15_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 7.1.4.105 `_ex15`

```
const ex GiNaC::_ex15 = ex(*_num15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.106 `_num18_p`

```
const numeric * GiNaC::_num18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 7.1.4.107 `_ex18`

```
const ex GiNaC::_ex18 = ex(*_num18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.108** `_num20_p`

```
const numeric * GiNaC::_num20_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.4.109** `_ex20`

```
const ex GiNaC::_ex20 = ex(*_num20_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.110** `_num24_p`

```
const numeric * GiNaC::_num24_p
```

Referenced by [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

**7.1.4.111** `_ex24`

```
const ex GiNaC::_ex24 = ex(*_num24_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.112** `_num25_p`

```
const numeric * GiNaC::_num25_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.4.113** `_ex25`

```
const ex GiNaC::_ex25 = ex(*_num25_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**7.1.4.114** `_num30_p`

```
const numeric * GiNaC::_num30_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**7.1.4.115** `_ex30`

```
const ex GiNaC::_ex30 = ex(*_num30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.116 `_num48_p`

```
const numeric * GiNaC::_num48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 7.1.4.117 `_ex48`

```
const ex GiNaC::_ex48 = ex(*_num48_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.118 `_num60_p`

```
const numeric * GiNaC::_num60_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 7.1.4.119 `_ex60`

```
const ex GiNaC::_ex60 = ex(*_num60_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 7.1.4.120 `_num120_p`

```
const numeric * GiNaC::_num120_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 7.1.4.121 `_ex120`

```
const ex GiNaC::_ex120 = ex(*_num120_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

## 7.2 GiNaC::Gt\_detail Namespace Reference

### Classes

- class [TransformExpressionWithCache](#)
- class [pathintegral\\_term](#)
- class [pathintegral\\_term\\_list](#)



## Functions

- `std::vector< std::vector< int > >` [integer\\_partition](#) (const int n, const int m)
- `template<typename Kernel>`  
[ex deconcatenate\\_path](#) (const std::vector< Kernel > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< Kernel > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `template ex deconcatenate\_path< ex >` (const std::vector< [ex](#) > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< [ex](#) > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `template ex deconcatenate\_path< Gt::kernel >` (const std::vector< [Gt::kernel](#) > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< [Gt::kernel](#) > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `std::ostream &` [operator<<](#) (std::ostream &os, const [pathintegral\\_term](#) &term)
- `bool` [operator<](#) (const [pathintegral\\_term](#) &a, const [pathintegral\\_term](#) &b)

## 7.2.1 Function Documentation

### 7.2.1.1 integer\_partition()

```
std::vector< std::vector< int > > GiNaC::Gt_detail::integer_partition (
    const int n,
    const int m)
```

Referenced by [deconcatenate\\_path\(\)](#).

### 7.2.1.2 deconcatenate\_path()

```
template<typename Kernel>
ex GiNaC::Gt_detail::deconcatenate_path (
    const std::vector< Kernel > & args,
    const std::vector< ex > & endpoints,
    const std::function< ex(std::vector< Kernel > &new_args, const ex &start, const
ex &end)> & construct)
```

References [integer\\_partition\(\)](#).

Referenced by [GiNaC::Gt::applyIntegrationPath\(\)](#).

### 7.2.1.3 deconcatenate\_path< ex >()

```
template ex GiNaC::Gt_detail::deconcatenate_path< ex > (
    const std::vector< ex > & args,
    const std::vector< ex > & endpoints,
    const std::function< ex(std::vector< ex > &new_args, const ex &start, const ex
&end)> & construct)
```

Referenced by [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#).

#### 7.2.1.4 deconcatenate\_path< Gt::kernel >()

```
template< ex GiNaC::Gt_detail::deconcatenate_path< Gt::kernel > (
    const std::vector< Gt::kernel > & args,
    const std::vector< ex > & endpoints,
    const std::function< ex(std::vector< Gt::kernel > &new_args, const ex &start,
const ex &end)> & construct)
```

#### 7.2.1.5 operator<<()

```
std::ostream & GiNaC::Gt_detail::operator<< (
    std::ostream & os,
    const pathintegral_term & term)
```

#### 7.2.1.6 operator<()

```
bool GiNaC::Gt_detail::operator< (
    const pathintegral_term & a,
    const pathintegral_term & b)
```

## 7.3 GiNaC::internal Namespace Reference

### Classes

- struct [\\_iter\\_rep](#)

## 7.4 std Namespace Reference

### Classes

- struct [hash< GiNaC::ex >](#)  
*Specialization of std::hash() for ex objects.*
- struct [equal\\_to< GiNaC::ex >](#)  
*Specialization of std::equal\_to() for ex objects.*
- struct [less< GiNaC::ptr< T > >](#)  
*Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.*

### Functions

- template<> void [swap](#) (GiNaC::ex &a, GiNaC::ex &b)  
*Specialization of [std::swap\(\)](#) for ex objects.*

## 7.4.1 Function Documentation

### 7.4.1.1 swap()

```
template<>
void std::swap (
    GiNaC::ex & a,
    GiNaC::ex & b) [inline]
```

Specialization of [std::swap\(\)](#) for ex objects.

References [GiNaC::ex::swap\(\)](#).

Referenced by [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), and [GiNaC::permutation\\_sign\(\)](#).



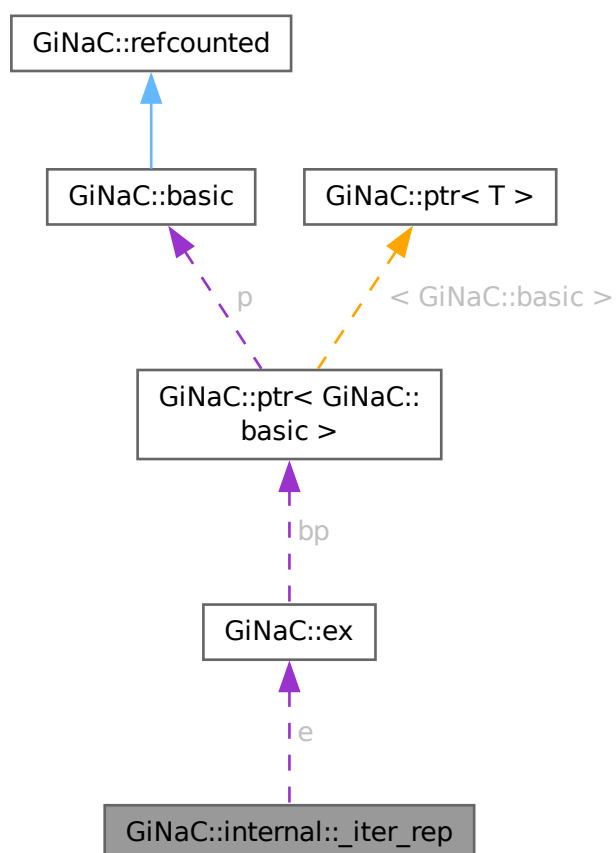
## Chapter 8

# Class Documentation

### 8.1 GiNaC::internal::\_iter\_rep Struct Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::internal::\_iter\_rep:



## Public Member Functions

- [\\_iter\\_rep](#) (const [ex](#) &[e\\_](#), [size\\_t](#) [i\\_](#), [size\\_t](#) [i\\_end\\_](#))
- bool [operator==](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept
- bool [operator!=](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept

## Public Attributes

- [ex](#) [e](#)
- [size\\_t](#) [i](#)
- [size\\_t](#) [i\\_end](#)

## 8.1.1 Constructor & Destructor Documentation

### 8.1.1.1 [\\_iter\\_rep](#)()

```
GiNaC::internal::_iter_rep::_iter_rep (
    const ex & e\_,
    size\_t i\_,
    size\_t i\_end\_) [inline]
```

References [e](#), [i](#), and [i\\_end](#).

Referenced by [operator!=\(\)](#), and [operator==\(\)](#).

## 8.1.2 Member Function Documentation

### 8.1.2.1 [operator==\(\)](#)

```
bool GiNaC::internal::_iter_rep::operator== (
    const \_iter\_rep & other) const [inline], [noexcept]
```

References [\\_iter\\_rep\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [e](#), and [i](#).

### 8.1.2.2 [operator!=\(\)](#)

```
bool GiNaC::internal::_iter_rep::operator!= (
    const \_iter\_rep & other) const [inline], [noexcept]
```

References [\\_iter\\_rep\(\)](#).

## 8.1.3 Member Data Documentation

### 8.1.3.1 [e](#)

```
ex GiNaC::internal::_iter_rep::e
```

Referenced by [\\_iter\\_rep\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 8.1.3.2 i

```
size_t GiNaC::internal::_iter_rep::i
```

Referenced by [\\_iter\\_rep\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 8.1.3.3 i\_end

```
size_t GiNaC::internal::_iter_rep::i_end
```

Referenced by [\\_iter\\_rep\(\)](#), and [GiNaC::const\\_preorder\\_iterator::increment\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 8.2 GiNaC::\_numeric\_digits Class Reference

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

```
#include <numeric.h>
```

### Public Member Functions

- [\\_numeric\\_digits](#) ()  
*\_numeric\_digits default ctor, checking for singleton invariance.*
- [\\_numeric\\_digits & operator=](#) (long prec)  
*Assign a native long to global Digits object.*
- [operator long](#) ()  
*Convert global Digits object to native type long.*
- void [print](#) (std::ostream &os) const  
*Append global Digits object to ostream.*
- void [add\\_callback](#) ([digits\\_changed\\_callback](#) callback)  
*Add a new callback function.*

### Private Attributes

- long [digits](#)  
*Number of decimal digits.*
- std::vector< [digits\\_changed\\_callback](#) > [callbacklist](#)

### Static Private Attributes

- static bool [too\\_late](#) = false  
*Already one object present.*

## 8.2.1 Detailed Description

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

We need an object rather than a dumbier basic type since as a side-effect we let it change `cl_default_float_format` when it gets changed. The only other meaningful thing to do with it is converting it to an unsigned, for temporarily storing its value e.g. The user must not create an own working object of this class! Since C++ forces us to make the class definition visible in order to use an object we put in a flag which prevents other objects of that class to be created.

## 8.2.2 Constructor & Destructor Documentation

### 8.2.2.1 `_numeric_digits()`

```
GiNaC::_numeric_digits::_numeric_digits ()
```

`_numeric_digits` default ctor, checking for singleton invariance.

References `digits`, and `too_late`.

Referenced by `operator=()`.

## 8.2.3 Member Function Documentation

### 8.2.3.1 `operator=()`

```
_numeric_digits & GiNaC::_numeric_digits::operator= (
    long prec)
```

Assign a native long to global Digits object.

References `_numeric_digits()`, `callbacklist`, and `digits`.

### 8.2.3.2 `operator long()`

```
GiNaC::_numeric_digits::operator long ()
```

Convert global Digits object to native type long.

References `digits`.

### 8.2.3.3 `print()`

```
void GiNaC::_numeric_digits::print (
    std::ostream & os) const
```

Append global Digits object to ostream.

References `digits`.

Referenced by `GiNaC::operator<<()`.



#### 8.2.3.4 add\_callback()

```
void GiNaC::_numeric_digits::add_callback (
    digits_changed_callback callback)
```

Add a new callback function.

References [callbacklist](#).

### 8.2.4 Member Data Documentation

#### 8.2.4.1 digits

```
long GiNaC::_numeric_digits::digits [private]
```

Number of decimal digits.

Referenced by [\\_numeric\\_digits\(\)](#), [operator long\(\)](#), [operator=\(\)](#), and [print\(\)](#).

#### 8.2.4.2 too\_late

```
bool GiNaC::_numeric_digits::too_late = false [static], [private]
```

Already one object present.

Referenced by [\\_numeric\\_digits\(\)](#).

#### 8.2.4.3 callbacklist

```
std::vector<digits_changed_callback> GiNaC::_numeric_digits::callbacklist [private]
```

Referenced by [add\\_callback\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following files:

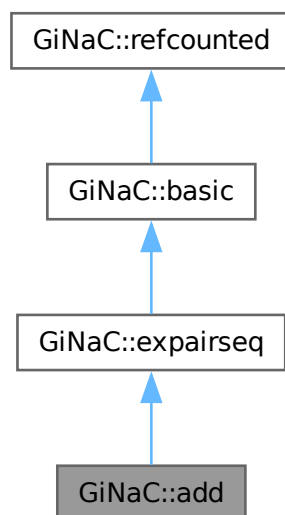
- [numeric.h](#)
- [numeric.cpp](#)

### 8.3 GiNaC::add Class Reference

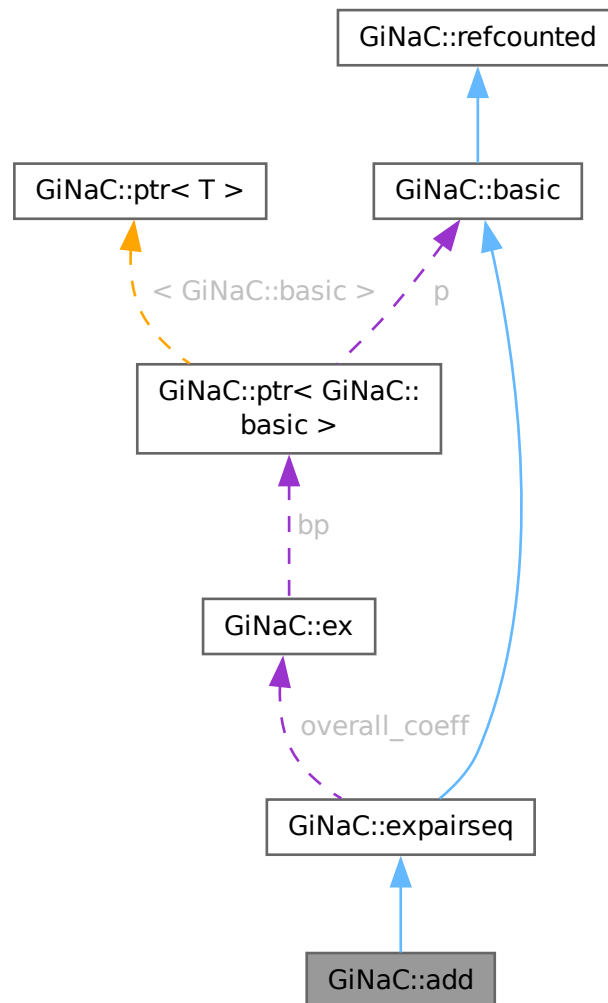
Sum of expressions.

```
#include <add.h>
```

Inheritance diagram for GiNaC::add:



Collaboration diagram for GiNaC::add:



### Public Member Functions

- `add` (const `ex` &lh, const `ex` &rh)
- `add` (const `exvector` &v)
- `add` (const `epvector` &v)
- `add` (const `epvector` &v, const `ex` &oc)
- `add` (`epvector` &&v)
- `add` (`epvector` &&v, const `ex` &oc)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*

- `int degree (const ex &s)` const override  
*Return degree of highest power in object s.*
- `int ldegree (const ex &s)` const override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1)` const override  
*Return coefficient of degree n in object s.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &r, int order, unsigned options=0)` const override  
*Implementation of `ex::series()` for sums.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const override  
*Implementation of `ex::normal()` for a sum.*
- `numeric integer_content ()` const override
- `ex smod (const numeric &xi)` const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient ()` const override  
*Implementation `ex::max_coefficient()`.*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- `ex eval_ncmul (const exvector &v)` const override

## Public Member Functions inherited from `GiNaC::expairseq`

- `expairseq (const ex &lh, const ex &rh)`
- `expairseq (const exvector &v)`
- `expairseq (const epvector &v, const ex &oc, bool do_index_renaming=false)`
- `expairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false)`
- `unsigned precedence ()` const override  
*Return relative operator precedence (for parenthezing output).*
- `bool info (unsigned inf)` const override  
*Information about the object.*
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex map (map_function &f)` const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval ()` const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational (exmap &repl)` const override  
*Implementation of `ex::to_rational()` for expairseqs.*
- `ex to_polynomial (exmap &repl)` const override  
*Implementation of `ex::to_polynomial()` for expairseqs.*
- `bool match (const ex &pattern, exmap &repl_lst)` const override  
*Check whether the expression matches a given pattern.*
- `ex subs (const exmap &m, unsigned options=0)` const override

*Substitute a set of objects by arbitrary expressions.*

- `ex conjugate ()` const override
- `void archive (archive_node &n)` const override
- *Save (serialize) the object into archive node.*
- `void read_archive (const archive_node &n, lst &symbols)` override
- *Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from GiNaC::basic

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate ()` const  
*Create a clone of this object on the heap.*
- `virtual ex evalf ()` const  
*Evaluate object numerically.*
- `virtual ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- `virtual void dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- `virtual ex operator[] (const ex &index)` const
- `virtual ex operator[] (size_t i)` const
- `virtual ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `virtual ex & operator[] (const ex &index)`
- `virtual ex & operator[] (size_t i)`
- `virtual bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- `virtual void accept (GiNaC::visitor &v)` const
- `virtual ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- `virtual ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- `virtual ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- `virtual bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- `template<class T>`  
`void print_dispatch (const print_context &c, unsigned level)` const  
*Like print(), but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like print(), but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options)` const

- *Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of `ex::diff()` for a sum.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex thisexpairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false) const override  
*Create an object of this type.*
- [ex thisexpairseq](#) ([epvector](#) &&vp, const [ex](#) &oc, bool do\_index\_renaming=false) const override
- [expair split\\_ex\\_to\\_pair](#) (const [ex](#) &e) const override  
*Form an expair from an ex, using the corresponding semantics.*
- [expair combine\\_ex\\_with\\_coeff\\_to\\_pair](#) (const [ex](#) &e, const [ex](#) &c) const override
- [expair combine\\_pair\\_with\\_coeff\\_to\\_pair](#) (const [expair](#) &p, const [ex](#) &c) const override
- [ex recombine\\_pair\\_to\\_ex](#) (const [expair](#) &p) const override  
*Form an ex out of an expair, using the corresponding semantics.*
- [ex expand](#) (unsigned options=0) const override  
*Expand expression, i.e.*
- void [print\\_add](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, const char \*mul\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::expairseq](#)

- [bool is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [return\\_type](#) () const override
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex expand](#) (unsigned options=0) const override  
*Expand expression, i.e.*
- virtual void [printseq](#) (const [print\\_context](#) &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void [printpair](#) (const [print\\_context](#) &c, const [expair](#) &p, unsigned upper\_precedence) const
- virtual bool [expair\\_needs\\_further\\_processing](#) ([epp](#) it)
- virtual [ex default\\_overall\\_coeff](#) () const
- virtual void [combine\\_overall\\_coeff](#) (const [ex](#) &c)
- virtual void [combine\\_overall\\_coeff](#) (const [ex](#) &c1, const [ex](#) &c2)
- virtual bool [can\\_make\\_flat](#) (const [expair](#) &p) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [construct\\_from\\_2\\_ex](#) (const [ex](#) &lh, const [ex](#) &rh)
- void [construct\\_from\\_2\\_expairseq](#) (const [expairseq](#) &s1, const [expairseq](#) &s2)
- void [construct\\_from\\_expairseq\\_ex](#) (const [expairseq](#) &s, const [ex](#) &e)
- void [construct\\_from\\_exvector](#) (const [exvector](#) &v)
- void [construct\\_from\\_epvector](#) (const [epvector](#) &v, bool do\_index\_renaming=false)
- void [construct\\_from\\_epvector](#) ([epvector](#) &&v, bool do\_index\_renaming=false)
- void [make\\_flat](#) (const [exvector](#) &v)  
*Combine this expairseq with argument exvector.*
- void [make\\_flat](#) (const [epvector](#) &v, bool do\_index\_renaming=false)  
*Combine this expairseq with argument epvector.*
- void [canonicalize](#) ()  
*Brings this expairseq into a sorted (canonical) form.*
- void [combine\\_same\\_terms\\_sorted\\_seq](#) ()  
*Compact a presorted expairseq by combining all matching expires to one each.*
- bool [is\\_canonical](#) () const  
*Check if this expairseq is in sorted (canonical) form.*
- [epvector expandchildren](#) (unsigned options) const  
*Member-wise expand the expires in this sequence.*
- [epvector evalchildren](#) () const  
*Member-wise evaluate the expires in this sequence.*
- [epvector subschildren](#) (const [exmap](#) &m, unsigned options=0) const  
*Member-wise substitute in this sequence.*

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

*Default output to stream.*

- void `do_print_tree` (const `print_tree` &c, unsigned level) const

*Tree output to stream.*

- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

*Python parsable output to stream.*

## Friends

- class `mul`
- class `power`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::expairseq`

- `epvector seq`
- `ex overall_coeff`

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 8.3.1 Detailed Description

Sum of expressions.

## 8.3.2 Constructor & Destructor Documentation

### 8.3.2.1 `add()` [1/6]

```
GiNaC::add::add (
    const ex & lh,
    const ex & rh)
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_2_ex()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_canonical()`, and `GiNaC::expairseq::overall_coeff`.

Referenced by `conjugate()`, and `power`.

### 8.3.2.2 `add()` [2/6]

```
GiNaC::add::add (
    const exvector & v)
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_exvector()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_canonical()`, and `GiNaC::expairseq::overall_coeff`.



**8.3.2.3 add() [3/6]**

```
GiNaC::add::add (
    const epvector & v)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.3.2.4 add() [4/6]**

```
GiNaC::add::add (
    const epvector & v,
    const ex & oc)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.3.2.5 add() [5/6]**

```
GiNaC::add::add (
    epvector && v)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.3.2.6 add() [6/6]**

```
GiNaC::add::add (
    epvector && v,
    const ex & oc)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.3.3 Member Function Documentation****8.3.3.1 precedence()**

```
unsigned GiNaC::add::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

References [precedence\(\)](#).

Referenced by [do\\_print\\_csrc\(\)](#), [precedence\(\)](#), and [print\\_add\(\)](#).

### 8.3.3.2 info()

```
bool GiNaC::add::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [info\(\)](#).

### 8.3.3.3 is\_polynomial()

```
bool GiNaC::add::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

### 8.3.3.4 degree()

```
int GiNaC::add::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object *s*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 8.3.3.5 ldegree()

```
int GiNaC::add::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object *s*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 8.3.3.6 coeff()

```
ex GiNaC::add::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

Referenced by [print\\_add\(\)](#), and [smod\(\)](#).

### 8.3.3.7 eval()

```
ex GiNaC::add::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x stands for a symbolic variables of type ex and c stands for such an expression that contain a plain number.

- $+(;c) \rightarrow c$
- $+(x;0) \rightarrow x$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::dynallocate\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluate](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

### 8.3.3.8 evalm()

```
ex GiNaC::add::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::matrix::add\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::evalm\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.3.3.9 series()

```
ex GiNaC::add::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for sums.

This performs series addition when adding pseries objects.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 8.3.3.10 normal()

```
ex GiNaC::add::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a sum.

It expands terms and performs fractional addition.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 8.3.3.11 integer\_content()

```
numeric GiNaC::add::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::denom\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

**8.3.3.12 smod()**

```
ex GiNaC::add::smod (
    const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

<i>xi</i>	modulus
-----------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

**8.3.3.13 max\_coefficient()**

```
numeric GiNaC::add::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

**8.3.3.14 conjugate()**

```
ex GiNaC::add::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [conjugate\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

Referenced by [conjugate\(\)](#).

### 8.3.3.15 real\_part()

```
ex GiNaC::add::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.3.3.16 imag\_part()

```
ex GiNaC::add::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.3.3.17 get\_free\_indices()

```
exvector GiNaC::add::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indices\\_consistent\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

### 8.3.3.18 eval\_ncmul()

```
ex GiNaC::add::eval_ncmul (
    const exvector & v) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

### 8.3.3.19 derivative()

```
ex GiNaC::add::derivative (
    const symbol & y) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a sum.

It differentiates each term.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), and [GiNaC::expairseq::seq](#).

**8.3.3.20 return\_type()**

```
unsigned GiNaC::add::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), and [GiNaC::expairseq::seq](#).

**8.3.3.21 return\_type\_tinfo()**

```
return_type_t GiNaC::add::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::make\\_return\\_type\\_t\(\)](#), and [GiNaC::expairseq::seq](#).

**8.3.3.22 thisexpairseq() [1/2]**

```
ex GiNaC::add::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::dynallocate\(\)](#).

**8.3.3.23 thisexpairseq() [2/2]**

```
ex GiNaC::add::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::dynallocate\(\)](#).

### 8.3.3.24 split\_ex\_to\_pair()

```
expair GiNaC::add::split_ex_to_pair (
    const ex & e) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [mul](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [evalm\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 8.3.3.25 combine\_ex\_with\_coeff\_to\_pair()

```
expair GiNaC::add::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [likely](#), [mul](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 8.3.3.26 combine\_pair\_with\_coeff\_to\_pair()

```
expair GiNaC::add::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::expair::coeff](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [GiNaC::mul::eval\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

### 8.3.3.27 recombine\_pair\_to\_ex()

```
ex GiNaC::add::recombine_pair_to_ex (
    const expair & p) const [override], [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::expair::coeff](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [eval\(\)](#), [evalm\(\)](#), [GiNaC::power::expand\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [normal\(\)](#), and [real\\_part\(\)](#).



**8.3.3.28 expand()**

```
ex GiNaC::add::expand (
    unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::expairseq::overall\\_co](#) and [GiNaC::basic::setflag\(\)](#).

Referenced by [normal\(\)](#).

**8.3.3.29 print\_add()**

```
void GiNaC::add::print_add (
    const print_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    unsigned level) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::class\\_info< OPT >::first](#), [coeff\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::expairseq::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**8.3.3.30 do\_print()**

```
void GiNaC::add::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [print\\_add\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.3.3.31 do\_print\_latex()**

```
void GiNaC::add::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

References [print\\_add\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.3.3.32 do\_print\_csrc()

```
void GiNaC::add::do_print_csrc (
    const print\_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [precedence\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::print\\_context::s](#), and [GiNaC::expairseq::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.3.3.33 do\_print\_python\_repr()

```
void GiNaC::add::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.3.4 Friends And Related Symbol Documentation

### 8.3.4.1 mul

```
friend class mul [friend]
```

References [mul](#).

Referenced by [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [mul](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.3.4.2 power

```
friend class power [friend]
```

References [add\(\)](#), and [power](#).

Referenced by [power](#).

The documentation for this class was generated from the following files:

- [add.h](#)
- [add.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 8.4 GiNaC::archive Class Reference

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

```
#include <archive.h>
```

### Classes

- struct [archived\\_ex](#)  
*Archived expression descriptor.*

### Public Member Functions

- [archive](#) ()  
*Construct archive from expression using the default name "ex".*
- [~archive](#) ()
- [archive](#) (const [ex](#) &e)  
*Construct archive from expression using the specified name.*
- [archive\\_ex](#) (const [ex](#) &e, const char \*name)  
*Archive an expression.*
- [ex\\_unarchive\\_ex](#) (const [lst](#) &sym\_lst, const char \*name) const  
*Retrieve expression from archive by name.*
- [ex\\_unarchive\\_ex](#) (const [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve expression from archive by index.*
- [ex\\_unarchive\\_ex](#) (const [lst](#) &sym\_lst, std::string &name, unsigned index=0) const  
*Retrieve expression and its name from archive by index.*
- unsigned [num\\_expressions](#) () const  
*Return number of archived expressions.*
- const [archive\\_node](#) & [get\\_top\\_node](#) (unsigned index=0) const  
*Return reference to top node of an expression specified by index.*
- void [clear](#) ()  
*Clear all archived expressions.*
- [archive\\_node\\_id](#) [add\\_node](#) (const [archive\\_node](#) &n)  
*Add [archive\\_node](#) to archive if the corresponding expression is not already archived.*
- [archive\\_node](#) & [get\\_node](#) ([archive\\_node\\_id](#) id)  
*Retrieve [archive\\_node](#) by ID.*
- void [forget](#) ()  
*Delete cached unarchived expressions in all [archive\\_nodes](#) (mainly for debugging).*
- void [prinraw](#) (std::ostream &os) const  
*Print archive to stream in ugly raw format (for debugging).*
- [archive\\_atom](#) [atomize](#) (const std::string &s) const  
*Atomize a string (i.e.*
- const std::string & [unatomize](#) ([archive\\_atom](#) id) const  
*Unatomize a string (i.e.*

### Private Types

- typedef std::map< std::string, [archive\\_atom](#) >::const\_iterator [inv\\_at\\_cit](#)  
*The map of from strings to indices of the atoms vectors allows for faster archiving.*

## Private Attributes

- `std::vector< archive_node > nodes`  
*Vector of archived nodes.*
- `std::vector< archived_ex > exprs`  
*Vector of archived expression descriptors.*
- `std::vector< std::string > atoms`  
*Vector of atomized strings (using a vector allows faster unarchiving).*
- `std::map< std::string, archive_atom > inverse_atoms`
- `std::map< ex, archive_node_id, ex_is_less > exprtable`  
*Map of stored expressions to nodes for faster archiving.*

## Friends

- `std::ostream & operator<< (std::ostream &os, const archive &ar)`  
*Write archive to binary data stream.*
- `std::istream & operator>> (std::istream &is, archive &ar)`  
*Read archive from binary data stream.*

## 8.4.1 Detailed Description

This class holds archived versions of [GiNaC](#) expressions (class `ex`).

An archive can be constructed from an expression and then written to a stream; or it can be read from a stream and then unarchived, yielding back the expression. Archives can hold multiple expressions which can be referred to by name or index number. The main component of the archive class is a vector of `archive_nodes` which each store one object of class `basic` (or a derived class).

## 8.4.2 Member Typedef Documentation

### 8.4.2.1 `inv_at_cit`

```
typedef std::map<std::string,archive_atom>::const_iterator GiNaC::archive::inv_at_cit [private]
```

The map of from strings to indices of the atoms vectors allows for faster archiving.

## 8.4.3 Constructor & Destructor Documentation

### 8.4.3.1 `archive()` [1/3]

```
GiNaC::archive::archive () [inline]
```

Referenced by `operator<<`, and `operator>>`.

### 8.4.3.2 `~archive()`

```
GiNaC::archive::~~archive () [inline]
```

### 8.4.3.3 archive() [2/3]

```
GiNaC::archive::archive (
    const ex & e) [inline]
```

Construct archive from expression using the default name "ex".

References [archive\\_ex\(\)](#).

### 8.4.3.4 archive() [3/3]

```
GiNaC::archive::archive (
    const ex & e,
    const char * n) [inline]
```

Construct archive from expression using the specified name.

References [archive\\_ex\(\)](#).

## 8.4.4 Member Function Documentation

### 8.4.4.1 archive\_ex()

```
void GiNaC::archive::archive_ex (
    const ex & e,
    const char * name)
```

Archive an expression.

#### Parameters

<i>e</i>	the expression to be archived
<i>name</i>	name under which the expression is stored

References [add\\_node\(\)](#), [atomize\(\)](#), and [exprs](#).

Referenced by [archive\(\)](#), and [archive\(\)](#).

### 8.4.4.2 unarchive\_ex() [1/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    const char * name) const
```

Retrieve expression from archive by name.

#### Parameters

<i>sym_lst</i>	list of pre-defined symbols
----------------	-----------------------------

<i>name</i>	name of expression
-------------	--------------------

References [atomize\(\)](#), [exprs](#), and [nodes](#).

#### 8.4.4.3 unarchive\_ex() [2/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    unsigned index = 0) const
```

Retrieve expression from archive by index.

##### Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>index</i>	index of expression

##### See also

[count\\_expressions](#)

References [exprs](#), and [nodes](#).

#### 8.4.4.4 unarchive\_ex() [3/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    std::string & name,
    unsigned index = 0) const
```

Retrieve expression and its name from archive by index.

##### Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>name</i>	receives the name of the expression
<i>index</i>	index of expression

##### See also

[count\\_expressions](#)

References [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 8.4.4.5 num\_expressions()

```
unsigned GiNaC::archive::num_expressions () const
```

Return number of archived expressions.

References [exprs](#).

#### 8.4.4.6 get\_top\_node()

```
const archive_node & GiNaC::archive::get_top_node (
    unsigned index = 0) const
```

Return reference to top node of an expression specified by index.

References [exprs](#), and [nodes](#).

#### 8.4.4.7 clear()

```
void GiNaC::archive::clear ()
```

Clear all archived expressions.

References [atoms](#), [exprs](#), [exprtable](#), [inverse\\_atoms](#), and [nodes](#).

#### 8.4.4.8 add\_node()

```
archive_node_id GiNaC::archive::add_node (
    const archive_node & n)
```

Add [archive\\_node](#) to archive if the corresponding expression is not already archived.

##### Returns

ID of archived node

References [exprtable](#), [GiNaC::archive\\_node::get\\_ex\(\)](#), [GiNaC::archive\\_node::has\\_ex\(\)](#), and [nodes](#).

Referenced by [archive\\_ex\(\)](#).

#### 8.4.4.9 get\_node()

```
archive_node & GiNaC::archive::get_node (
    archive_node_id id)
```

Retrieve [archive\\_node](#) by ID.

References [nodes](#).

#### 8.4.4.10 forget()

```
void GiNaC::archive::forget ()
```

Delete cached unarchived expressions in all archive\_nodes (mainly for debugging).

References [GiNaC::archive\\_node::forget\(\)](#), and [nodes](#).

#### 8.4.4.11 printraw()

```
void GiNaC::archive::printraw (
    std::ostream & os) const
```

Print archive to stream in ugly raw format (for debugging).

References [atoms](#), [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 8.4.4.12 atomize()

```
archive_atom GiNaC::archive::atomize (
    const std::string & s) const
```

Atomize a string (i.e.

convert it into an ID number that uniquely represents the string).

References [atoms](#), and [inverse\\_atoms](#).

Referenced by [archive\\_ex\(\)](#), and [unarchive\\_ex\(\)](#).

#### 8.4.4.13 unatomize()

```
const std::string & GiNaC::archive::unatomize (
    archive_atom id) const
```

Unatomize a string (i.e.

convert the ID number back to the string).

References [atoms](#).

Referenced by [printraw\(\)](#), and [unarchive\\_ex\(\)](#).

### 8.4.5 Friends And Related Symbol Documentation

#### 8.4.5.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const archive & ar) [friend]
```

Write archive to binary data stream.

References [archive\(\)](#), [atoms](#), [exprs](#), [GINACLIB\\_ARCHIVE\\_VERSION](#), [nodes](#), and [GiNaC::write\\_unsigned\(\)](#).



### 8.4.5.2 operator>>

```
std::istream & operator>> (  
    std::istream & is,  
    archive & ar) [friend]
```

Read archive from binary data stream.

References [archive\(\)](#), [atoms](#), [exprs](#), [GINACLIB\\_ARCHIVE\\_AGE](#), [GINACLIB\\_ARCHIVE\\_VERSION](#), [inverse\\_atoms](#), [nodes](#), and [GiNaC::read\\_unsigned\(\)](#).

## 8.4.6 Member Data Documentation

### 8.4.6.1 nodes

```
std::vector<archive_node> GiNaC::archive::nodes [private]
```

Vector of archived nodes.

Referenced by [add\\_node\(\)](#), [clear\(\)](#), [forget\(\)](#), [get\\_node\(\)](#), [get\\_top\\_node\(\)](#), [operator<<](#), [operator>>](#), [prinraw\(\)](#), [unarchive\\_ex\(\)](#), [unarchive\\_ex\(\)](#), and [unarchive\\_ex\(\)](#).

### 8.4.6.2 exprs

```
std::vector<archived_ex> GiNaC::archive::exprs [private]
```

Vector of archived expression descriptors.

Referenced by [archive\\_ex\(\)](#), [clear\(\)](#), [get\\_top\\_node\(\)](#), [num\\_expressions\(\)](#), [operator<<](#), [operator>>](#), [prinraw\(\)](#), [unarchive\\_ex\(\)](#), [unarchive\\_ex\(\)](#), and [unarchive\\_ex\(\)](#).

### 8.4.6.3 atoms

```
std::vector<std::string> GiNaC::archive::atoms [mutable], [private]
```

Vector of atomized strings (using a vector allows faster unarchiving).

Referenced by [atomize\(\)](#), [clear\(\)](#), [operator<<](#), [operator>>](#), [prinraw\(\)](#), and [unatomize\(\)](#).

### 8.4.6.4 inverse\_atoms

```
std::map<std::string, archive_atom> GiNaC::archive::inverse_atoms [mutable], [private]
```

Referenced by [atomize\(\)](#), [clear\(\)](#), and [operator>>](#).

#### 8.4.6.5 exprtable

```
std::map<ex, archive_node_id, ex_is_less> GiNaC::archive::exprtable [mutable], [private]
```

Map of stored expressions to nodes for faster archiving.

Referenced by [add\\_node\(\)](#), and [clear\(\)](#).

The documentation for this class was generated from the following files:

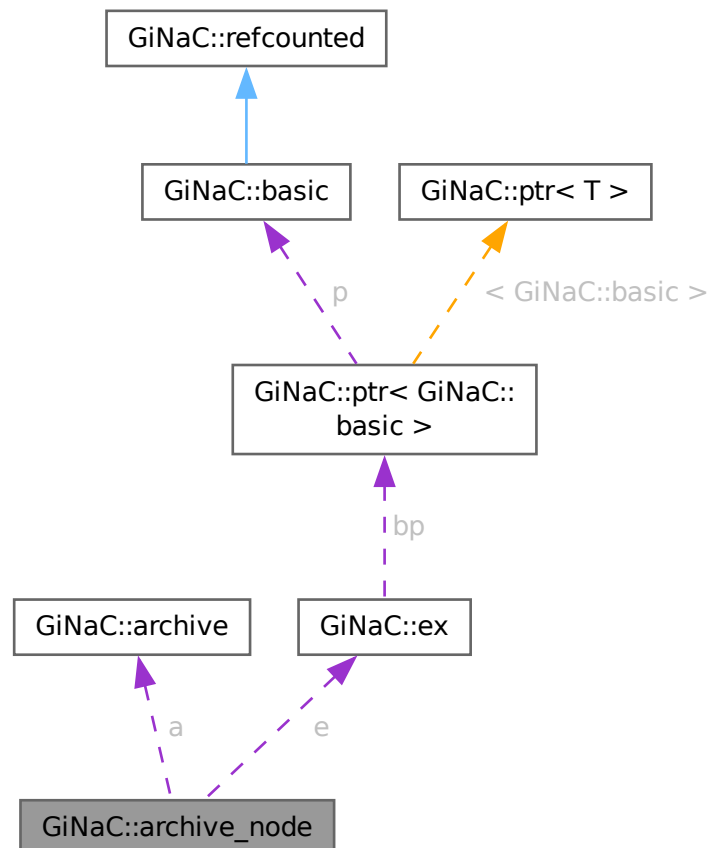
- [archive.h](#)
- [archive.cpp](#)

### 8.5 GiNaC::archive\_node Class Reference

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node:



## Classes

- struct [property\\_info](#)  
*Information about a stored property.*
- struct [property](#)  
*Archived property (data type, name and associated data).*
- struct [archive\\_node\\_cit\\_range](#)

## Public Types

- enum [property\\_type](#) { [PTYPE\\_BOOL](#) , [PTYPE\\_UNSIGNED](#) , [PTYPE\\_STRING](#) , [PTYPE\\_NODE](#) }  
*Property data types.*
- typedef std::vector< [property\\_info](#) > [propinfovector](#)
- typedef std::vector< [property](#) >::const\_iterator [archive\\_node\\_cit](#)

## Public Member Functions

- [archive\\_node](#) ([archive](#) &ar)
- [archive\\_node](#) ([archive](#) &ar, const [ex](#) &expr)  
*Recursively construct archive node from expression.*
- const [archive\\_node](#) & [operator=](#) (const [archive\\_node](#) &other)  
*Assignment operator of [archive\\_node](#).*
- void [add\\_bool](#) (const std::string &name, bool value)  
*Add property of type "bool" to node.*
- void [add\\_unsigned](#) (const std::string &name, unsigned value)  
*Add property of type "unsigned int" to node.*
- void [add\\_string](#) (const std::string &name, const std::string &value)  
*Add property of type "string" to node.*
- void [add\\_ex](#) (const std::string &name, const [ex](#) &value)  
*Add property of type "ex" to node.*
- bool [find\\_bool](#) (const std::string &name, bool &ret, unsigned index=0) const  
*Retrieve property of type "bool" from node.*
- bool [find\\_unsigned](#) (const std::string &name, unsigned &ret, unsigned index=0) const  
*Retrieve property of type "unsigned" from node.*
- bool [find\\_string](#) (const std::string &name, std::string &ret, unsigned index=0) const  
*Retrieve property of type "string" from node.*
- [archive\\_node\\_cit](#) [find\\_first](#) (const std::string &name) const  
*Find the location in the vector of properties of the first/last property with a given name.*
- [archive\\_node\\_cit](#) [find\\_last](#) (const std::string &name) const
- [archive\\_node\\_cit\\_range](#) [find\\_property\\_range](#) (const std::string &name1, const std::string &name2) const  
*Find a range of locations in the vector of properties.*
- bool [find\\_ex](#) (const std::string &name, [ex](#) &ret, [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve property of type "ex" from node.*
- void [find\\_ex\\_by\\_loc](#) ([archive\\_node\\_cit](#) loc, [ex](#) &ret, [lst](#) &sym\_lst) const  
*Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.*
- const [archive\\_node](#) & [find\\_ex\\_node](#) (const std::string &name, unsigned index=0) const  
*Retrieve property of type "ex" from node, returning the node of the sub-expression.*
- void [get\\_properties](#) ([propinfovector](#) &v) const  
*Return vector of properties stored in node.*

- `ex unarchive (lst &sym_lst) const`  
Convert *archive node* to *GiNaC expression*.
- `bool has_same_ex_as (const archive_node &other) const`  
Check if the *archive\_node* stores the same expression as another *archive\_node*.
- `bool has_ex () const`
- `ex get_ex () const`
- `void forget ()`  
Delete cached unarchived expressions from node (for debugging).
- `void printraw (std::ostream &os) const`  
Output *archive\_node* to stream in ugly raw format (for debugging).

### Private Attributes

- `archive & a`  
Reference to the archive to which this node belongs.
- `std::vector< property > props`  
Vector of stored properties.
- `bool has_expression`  
Flag indicating whether a cached unarchived representation of this node exists.
- `ex e`  
The cached unarchived representation of this node (if any).

### Friends

- `std::ostream & operator<< (std::ostream &os, const archive_node &ar)`  
Write *archive\_node* to binary data stream.
- `std::istream & operator>> (std::istream &is, archive_node &ar)`  
Read *archive\_node* from binary data stream.

## 8.5.1 Detailed Description

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class). Each property is addressed by its name and data type.

## 8.5.2 Member Typedef Documentation

### 8.5.2.1 propinfovector

```
typedef std::vector<property_info> GiNaC::archive_node::propinfovector
```

### 8.5.2.2 archive\_node\_cit

```
typedef std::vector<property>::const_iterator GiNaC::archive_node::archive_node_cit
```

## 8.5.3 Member Enumeration Documentation

### 8.5.3.1 property\_type

```
enum GiNaC::archive_node::property_type
```

Property data types.

#### Enumerator

PTYPE_BOOL	
------------	--

PTYPE_UNSIGNED	
PTYPE_STRING	
PTYPE_NODE	

## 8.5.4 Constructor & Destructor Documentation

### 8.5.4.1 archive\_node() [1/2]

```
GiNaC::archive_node::archive_node (
    archive & ar) [inline]
```

References [a](#), and [has\\_expression](#).

Referenced by [add\\_ex\(\)](#), [find\\_ex\\_node\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator<<](#), [operator=\(\)](#), and [operator>>](#).

### 8.5.4.2 archive\_node() [2/2]

```
GiNaC::archive_node::archive_node (
    archive & ar,
    const ex & expr)
```

Recursively construct archive node from expression.

References [a](#), [GiNaC::ex::bp](#), [e](#), and [has\\_expression](#).

## 8.5.5 Member Function Documentation

### 8.5.5.1 operator=()

```
const archive_node & GiNaC::archive_node::operator= (
    const archive_node & other)
```

Assignment operator of [archive\\_node](#).

References [archive\\_node\(\)](#), [e](#), [has\\_expression](#), and [props](#).

### 8.5.5.2 add\_bool()

```
void GiNaC::archive_node::add_bool (
    const std::string & name,
    bool value)
```

Add property of type "bool" to node.

References [a](#), [props](#), and [PTYPE\\_BOOL](#).

Referenced by [GiNaC::minkmetric::archive\(\)](#), [GiNaC::spinidx::archive\(\)](#), [GiNaC::tensepsilon::archive\(\)](#), and [GiNaC::varidx::archive\(\)](#).

### 8.5.5.3 add\_unsigned()

```
void GiNaC::archive_node::add_unsigned (
    const std::string & name,
    unsigned value)
```

Add property of type "unsigned int" to node.

References [a](#), [props](#), and [PTYPE\\_UNSIGNED](#).

Referenced by [GiNaC::clifford::archive\(\)](#), [GiNaC::color::archive\(\)](#), [GiNaC::fderivative::archive\(\)](#), [GiNaC::matrix::archive\(\)](#), [GiNaC::relational::archive\(\)](#), [GiNaC::symmetry::archive\(\)](#), and [GiNaC::wildcard::archive\(\)](#).

### 8.5.5.4 add\_string()

```
void GiNaC::archive_node::add_string (
    const std::string & name,
    const std::string & value)
```

Add property of type "string" to node.

References [a](#), [props](#), and [PTYPE\\_STRING](#).

Referenced by [GiNaC::basic::archive\(\)](#), [GiNaC::constant::archive\(\)](#), [GiNaC::function::archive\(\)](#), [GiNaC::numeric::archive\(\)](#), and [GiNaC::symbol::archive\(\)](#).

### 8.5.5.5 add\_ex()

```
void GiNaC::archive_node::add_ex (
    const std::string & name,
    const ex & value)
```

Add property of type "ex" to node.

References [a](#), [archive\\_node\(\)](#), [props](#), and [PTYPE\\_NODE](#).

Referenced by [GiNaC::clifford::archive\(\)](#), [GiNaC::container< std::vector >::archive\(\)](#), [GiNaC::expairseq::archive\(\)](#), [GiNaC::Gt::archive\(\)](#), [GiNaC::idx::archive\(\)](#), [GiNaC::indexed::archive\(\)](#), [GiNaC::integral::archive\(\)](#), [GiNaC::matrix::archive\(\)](#), [GiNaC::power::archive\(\)](#), [GiNaC::pseries::archive\(\)](#), [GiNaC::relational::archive\(\)](#), and [GiNaC::symmetry::archive\(\)](#).

### 8.5.5.6 find\_bool()

```
bool GiNaC::archive_node::find_bool (
    const std::string & name,
    bool & ret,
    unsigned index = 0) const
```

Retrieve property of type "bool" from node.

#### Returns

"true" if property was found, "false" otherwise

References [a](#), [props](#), and [PTYPE\\_BOOL](#).

Referenced by [GiNaC::minkmetric::read\\_archive\(\)](#), [GiNaC::spinidx::read\\_archive\(\)](#), [GiNaC::tensepsilon::read\\_archive\(\)](#), and [GiNaC::varidx::read\\_archive\(\)](#).

#### 8.5.5.7 find\_unsigned()

```
bool GiNaC::archive_node::find_unsigned (
    const std::string & name,
    unsigned & ret,
    unsigned index = 0) const
```

Retrieve property of type "unsigned" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [props](#), and [PTYPE\\_UNSIGNED](#).

Referenced by [GiNaC::clifford::read\\_archive\(\)](#), [GiNaC::color::read\\_archive\(\)](#), [GiNaC::fderivative::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::matrix::read\\_archive\(\)](#), [GiNaC::relational::read\\_archive\(\)](#), [GiNaC::symmetry::read\\_archive\(\)](#), and [GiNaC::wildcard::read\\_archive\(\)](#).

#### 8.5.5.8 find\_string()

```
bool GiNaC::archive_node::find_string (
    const std::string & name,
    std::string & ret,
    unsigned index = 0) const
```

Retrieve property of type "string" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [props](#), and [PTYPE\\_STRING](#).

Referenced by [GiNaC::constant::read\\_archive\(\)](#), [GiNaC::function::read\\_archive\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), and [unarchive\(\)](#).

#### 8.5.5.9 find\_first()

```
archive_node::archive_node_cit GiNaC::archive_node::find_first (
    const std::string & name) const
```

Find the location in the vector of properties of the first/last property with a given name.

References [a](#), and [props](#).

#### 8.5.5.10 find\_last()

```
archive_node::archive_node_cit GiNaC::archive_node::find_last (
    const std::string & name) const
```

References [a](#), and [props](#).

### 8.5.5.11 find\_property\_range()

```
archive_node::archive_node_cit_range GiNaC::archive_node::find_property_range (
    const std::string & name1,
    const std::string & name2) const
```

Find a range of locations in the vector of properties.

The result begins at the first property with name1 and ends one past the last property with name2.

References [a](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::begin](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::end](#), and [props](#).

Referenced by [GiNaC::container< std::vector >::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::matrix::read\\_archive\(\)](#), and [GiNaC::pseries::read\\_archive\(\)](#).

### 8.5.5.12 find\_ex()

```
bool GiNaC::archive_node::find_ex (
    const std::string & name,
    ex & ret,
    lst & sym_lst,
    unsigned index = 0) const
```

Retrieve property of type "ex" from node.

#### Returns

"true" if property was found, "false" otherwise

References [a](#), [props](#), and [PTYPE\\_NODE](#).

Referenced by [GiNaC::clifford::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::Gt::read\\_archive\(\)](#), [GiNaC::idx::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::integral::read\\_archive\(\)](#), [GiNaC::power::read\\_archive\(\)](#), [GiNaC::pseries::read\\_archive\(\)](#), [GiNaC::relational::read\\_archive\(\)](#), and [GiNaC::symmetry::read\\_archive\(\)](#).

### 8.5.5.13 find\_ex\_by\_loc()

```
void GiNaC::archive_node::find_ex_by_loc (
    archive_node_cit loc,
    ex & ret,
    lst & sym_lst) const
```

Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.

This is much more efficient than the preceding function.

References [a](#).

Referenced by [GiNaC::container< std::vector >::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::matrix::read\\_archive\(\)](#), and [GiNaC::pseries::read\\_archive\(\)](#).



#### 8.5.5.14 find\_ex\_node()

```
const archive_node & GiNaC::archive_node::find_ex_node (
    const std::string & name,
    unsigned index = 0) const
```

Retrieve property of type "ex" from node, returning the node of the sub-expression.

References [a](#), [archive\\_node\(\)](#), [props](#), and [PTYPE\\_NODE](#).

#### 8.5.5.15 get\_properties()

```
void GiNaC::archive_node::get_properties (
    propinfovector & v) const
```

Return vector of properties stored in node.

References [a](#), and [props](#).

#### 8.5.5.16 unarchive()

```
ex GiNaC::archive_node::unarchive (
    lst & sym_lst) const
```

Convert archive node to [GiNaC](#) expression.

References [GiNaC::status\\_flags::dynallocated](#), [e](#), [GiNaC::find\\_factory\\_fcn\(\)](#), [find\\_string\(\)](#), and [has\\_expression](#).

#### 8.5.5.17 has\_same\_ex\_as()

```
bool GiNaC::archive_node::has_same_ex_as (
    const archive_node & other) const
```

Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).

##### Returns

"true" if expressions are the same

References [archive\\_node\(\)](#), [GiNaC::ex::bp](#), [e](#), and [has\\_expression](#).

#### 8.5.5.18 has\_ex()

```
bool GiNaC::archive_node::has_ex () const [inline]
```

References [has\\_expression](#).

Referenced by [GiNaC::archive::add\\_node\(\)](#).

#### 8.5.5.19 `get_ex()`

```
ex GiNaC::archive_node::get_ex () const [inline]
```

References [e](#).

Referenced by [GiNaC::archive::add\\_node\(\)](#).

#### 8.5.5.20 `forget()`

```
void GiNaC::archive_node::forget ()
```

Delete cached unarchived expressions from node (for debugging).

References [e](#), and [has\\_expression](#).

Referenced by [GiNaC::archive::forget\(\)](#).

#### 8.5.5.21 `printraw()`

```
void GiNaC::archive_node::printraw (  
    std::ostream & os) const
```

Output [archive\\_node](#) to stream in ugly raw format (for debugging).

References [a](#), [e](#), [has\\_expression](#), [props](#), [PTYPE\\_BOOL](#), [PTYPE\\_NODE](#), [PTYPE\\_STRING](#), and [PTYPE\\_UNSIGNED](#).

### 8.5.6 Friends And Related Symbol Documentation

#### 8.5.6.1 `operator<<`

```
std::ostream & operator<< (  
    std::ostream & os,  
    const archive_node & ar) [friend]
```

Write [archive\\_node](#) to binary data stream.

References [archive\\_node\(\)](#), [props](#), and [GiNaC::write\\_unsigned\(\)](#).

#### 8.5.6.2 `operator>>`

```
std::istream & operator>> (  
    std::istream & is,  
    archive_node & ar) [friend]
```

Read [archive\\_node](#) from binary data stream.

References [archive\\_node\(\)](#), [props](#), and [GiNaC::read\\_unsigned\(\)](#).

## 8.5.7 Member Data Documentation

### 8.5.7.1 a

```
archive& GiNaC::archive_node::a [private]
```

Reference to the archive to which this node belongs.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [archive\\_node\(\)](#), [archive\\_node\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_by\\_loc\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), and [printraw\(\)](#).

### 8.5.7.2 props

```
std::vector<property> GiNaC::archive_node::props [private]
```

Vector of stored properties.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), [operator<<](#), [operator=\(\)](#), [operator>>](#), and [printraw\(\)](#).

### 8.5.7.3 has\_expression

```
bool GiNaC::archive_node::has_expression [mutable], [private]
```

Flag indicating whether a cached unarchived representation of this node exists.

Referenced by [archive\\_node\(\)](#), [archive\\_node\(\)](#), [forget\(\)](#), [has\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [printraw\(\)](#), and [unarchive\(\)](#).

### 8.5.7.4 e

```
ex GiNaC::archive_node::e [mutable], [private]
```

The cached unarchived representation of this node (if any).

Referenced by [archive\\_node\(\)](#), [forget\(\)](#), [get\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [printraw\(\)](#), and [unarchive\(\)](#).

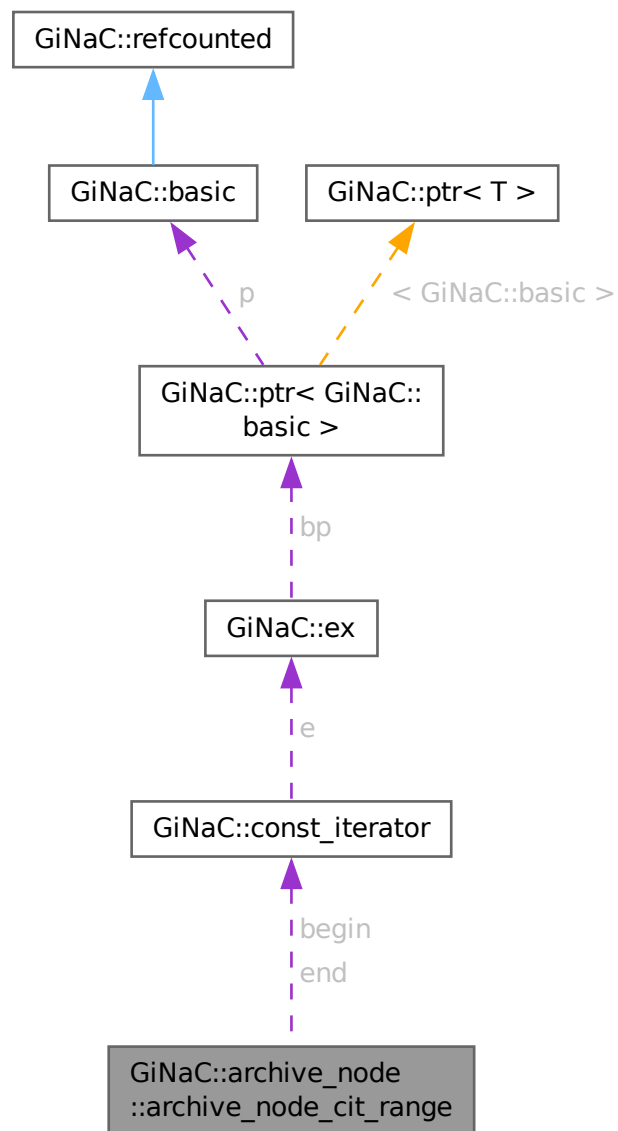
The documentation for this class was generated from the following files:

- [archive.h](#)
- [archive.cpp](#)

## 8.6 GiNaC::archive\_node::archive\_node\_cit\_range Struct Reference

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node::archive\_node\_cit\_range:



### Public Attributes

- [archive\\_node\\_cit begin](#)
- [archive\\_node\\_cit end](#)

## 8.6.1 Member Data Documentation

### 8.6.1.1 begin

`archive_node_cit` GiNaC::archive\_node::archive\_node\_cit\_range::begin

Referenced by [GiNaC::archive\\_node::find\\_property\\_range\(\)](#).

### 8.6.1.2 end

`archive_node_cit` GiNaC::archive\_node::archive\_node\_cit\_range::end

Referenced by [GiNaC::archive\\_node::find\\_property\\_range\(\)](#).

The documentation for this struct was generated from the following file:

- [archive.h](#)

## 8.7 GiNaC::archive::archived\_ex Struct Reference

Archived expression descriptor.

### Public Member Functions

- [archived\\_ex](#) ()
- [archived\\_ex](#) ([archive\\_atom](#) n, [archive\\_node\\_id](#) node)

### Public Attributes

- [archive\\_atom](#) name  
*Name of expression.*
- [archive\\_node\\_id](#) root  
*ID of root node.*

## 8.7.1 Detailed Description

Archived expression descriptor.

## 8.7.2 Constructor & Destructor Documentation

### 8.7.2.1 archived\_ex() [1/2]

GiNaC::archive::archived\_ex::archived\_ex () [inline]

### 8.7.2.2 `archived_ex()` [2/2]

```
GiNaC::archive::archived_ex::archived_ex (
    archive_atom n,
    archive_node_id node) [inline]
```

References [name](#), and [root](#).

## 8.7.3 Member Data Documentation

### 8.7.3.1 `name`

`archive_atom` GiNaC::archive::archived\_ex::name

Name of expression.

Referenced by [archived\\_ex\(\)](#).

### 8.7.3.2 `root`

`archive_node_id` GiNaC::archive::archived\_ex::root

ID of root node.

Referenced by [archived\\_ex\(\)](#).

The documentation for this struct was generated from the following file:

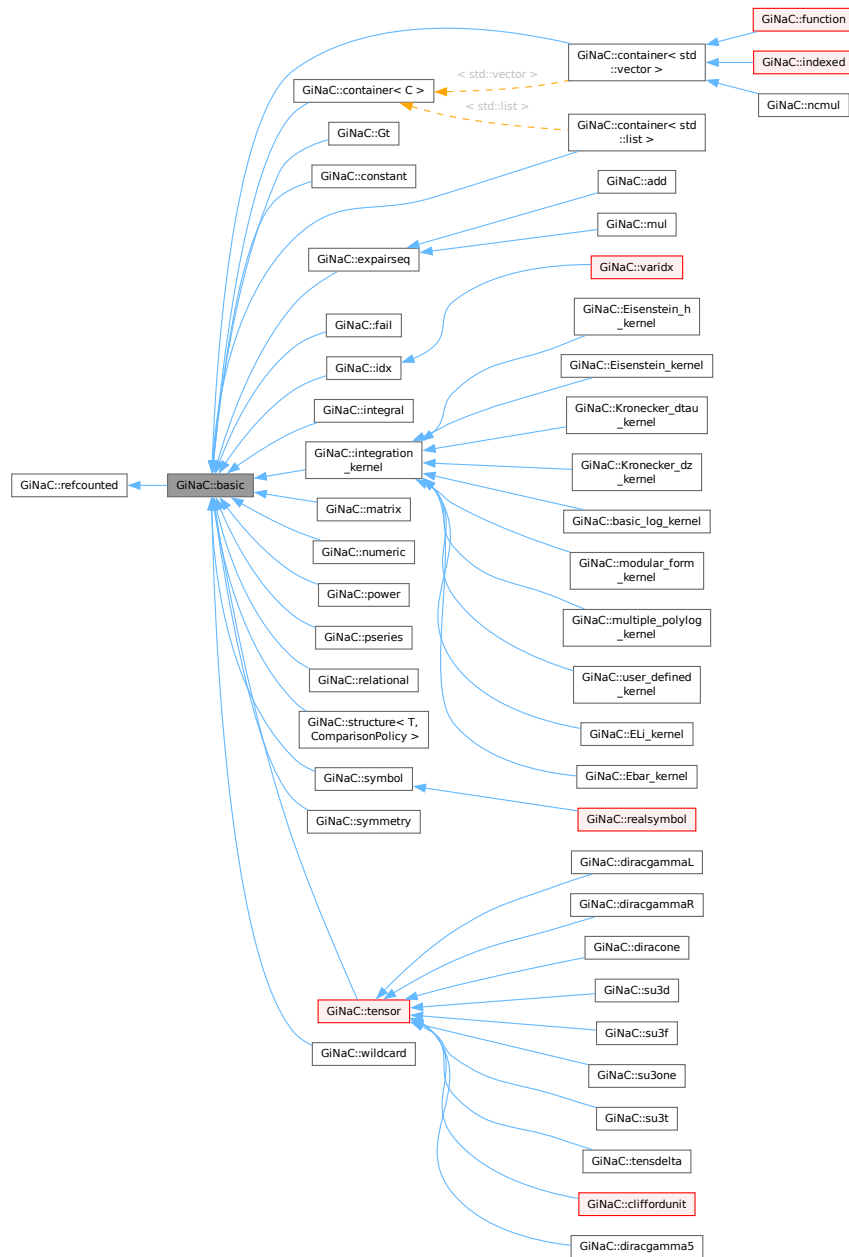
- [archive.h](#)

## 8.8 GiNaC::basic Class Reference

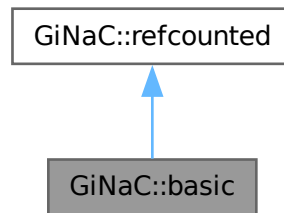
This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

```
#include <basic.h>
```

Inheritance diagram for GiNaC::basic:



Collaboration diagram for GiNaC::basic:



## Public Member Functions

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level=0`) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool `info` (unsigned `inf`) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t `i`) const  
*Return operand/member at position `i`.*
- virtual `ex operator[]` (const `ex` &`index`) const
- virtual `ex operator[]` (size\_t `i`) const



- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type () const`
- virtual `return_type_t return_type_tinfo () const`
- virtual `ex conjugate () const`
- virtual `ex real_part () const`
- virtual `ex imag_part () const`

- `template<class T>`  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Friends

- class [ex](#)

## 8.8.1 Detailed Description

This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

## 8.8.2 Constructor & Destructor Documentation

### 8.8.2.1 [basic\(\)](#) [1/2]

```
GiNaC::basic::basic () [inline], [protected]
```

References [basic\(\)](#), and [flags](#).

Referenced by [basic\(\)](#), [basic\(\)](#), [clearflag\(\)](#), [compare\(\)](#), [compare\\_same\\_type\(\)](#), [duplicate\(\)](#), [eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\\_indexed\(\)](#), [hold\(\)](#), [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_equal\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::constant::is\\_equal\\_same\\_type\(\)](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::container< std::vector >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::expairseq::is\\_equal\\_same\\_type\(\)](#), [GiNaC::numeric::is\\_equal\\_same\\_type\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::symbol::is\\_equal\\_same\\_type\(\)](#), [map\(\)](#), [match\\_same\\_type\(\)](#), [GiNaC::idx::match\\_same\\_type\(\)](#), [GiNaC::matrix::match\\_same\\_type\(\)](#), [GiNaC::relational::match\\_same\\_type\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::match\\_same\\_type\(\)](#), [operator=\(\)](#), [setflag\(\)](#), [subs\(\)](#), and [GiNaC::symmetry::sy\\_swap](#).

### 8.8.2.2 [~basic\(\)](#)

```
virtual GiNaC::basic::~~basic () [inline], [virtual]
```

[basic](#) destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).

References [GiNaC::status\\_flags::dynallocated](#), [flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GINAC\\_ASSERT](#).

### 8.8.2.3 [basic\(\)](#) [2/2]

```
GiNaC::basic::basic (
    const basic & other)
```

References [basic\(\)](#).

## 8.8.3 Member Function Documentation

### 8.8.3.1 operator=()

```
const basic & GiNaC::basic::operator= (
    const basic & other)
```

basic assignment operator: the other object might be of a derived class.

References [basic\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [hashvalue](#), [operator=\(\)](#), and [GiNaC::refcounted::set\\_refcount\(\)](#).

Referenced by [operator=\(\)](#).

### 8.8.3.2 duplicate()

```
virtual basic * GiNaC::basic::duplicate () const [inline], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented in [GiNaC::possymbol](#), and [GiNaC::realsymbol](#).

References [basic\(\)](#), [GiNaC::status\\_flags::dynamallocated](#), and [setflag\(\)](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), [map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), [subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::spinidx::toggle\\_dot\(\)](#), [GiNaC::varidx::toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

### 8.8.3.3 eval()

```
ex GiNaC::basic::eval () const [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

References [eval\(\)](#), [ex](#), and [hold\(\)](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), and [eval\(\)](#).

### 8.8.3.4 evalf()

```
ex GiNaC::basic::evalf () const [virtual]
```

Evaluate object numerically.

Reimplemented in [GiNaC::constant](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::symbol](#).

References [evalf\(\)](#), [ex](#), [map\(\)](#), and [nops\(\)](#).

Referenced by [evalf\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 8.8.3.5 evalm()

```
ex GiNaC::basic::evalm () const [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented in [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [evalm\(\)](#), [ex](#), [map\(\)](#), [GiNaC::map\\_evalm](#), and [nops\(\)](#).

Referenced by [evalm\(\)](#).

### 8.8.3.6 eval\_integ()

```
ex GiNaC::basic::eval_integ () const [virtual]
```

Evaluate integrals, if result is known.

Reimplemented in [GiNaC::integral](#), and [GiNaC::pseries](#).

References [eval\\_integ\(\)](#), [ex](#), [map\(\)](#), [GiNaC::map\\_eval\\_integ](#), and [nops\(\)](#).

Referenced by [eval\\_integ\(\)](#).

### 8.8.3.7 eval\_ncmul()

```
ex GiNaC::basic::eval_ncmul (
    const exvector & v) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [eval\\_ncmul\(\)](#), [ex](#), and [GiNaC::hold\\_ncmul\(\)](#).

Referenced by [eval\\_ncmul\(\)](#).

### 8.8.3.8 eval\_indexed()

```
ex GiNaC::basic::eval_indexed (
    const basic & i) const [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented in [GiNaC::matrix](#), [GiNaC::minkmetric](#), [GiNaC::spinmetric](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::su3d](#), [GiNaC::su3f](#), [GiNaC::tensdelta](#), [GiNaC::tensepsilon](#), and [GiNaC::tensmetric](#).

References [basic\(\)](#), [eval\\_indexed\(\)](#), and [ex](#).

Referenced by [eval\\_indexed\(\)](#).

### 8.8.3.9 print()

```
void GiNaC::basic::print (
    const print_context & c,
    unsigned level = 0) const [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

#### Parameters

<code>c</code>	print context object that describes the output formatting
----------------	---

<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting
--------------	---

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [print\(\)](#), and [print\\_dispatch\(\)](#).

Referenced by [dbgprint\(\)](#), [dbgprinttree\(\)](#), [print\(\)](#), [GiNaC::fderivative::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), and [GiNaC::print\\_sym\\_pow\(\)](#).

#### 8.8.3.10 dbgprint()

```
void GiNaC::basic::dbgprint () const [virtual]
```

Little wrapper around print to be called within a debugger.

This is needed because you cannot call `foo.print(cout)` from within the debugger because it might not know what `cout` is. This method can be invoked with no argument and it will simply print to `stdout`.

See also

[basic::print](#)  
[basic::dbgprinttree](#)

References [dbgprint\(\)](#), and [print\(\)](#).

Referenced by [dbgprint\(\)](#).

#### 8.8.3.11 dbgprinttree()

```
void GiNaC::basic::dbgprinttree () const [virtual]
```

Little wrapper around printtree to be called within a debugger.

See also

[basic::dbgprint](#)

References [dbgprinttree\(\)](#), and [print\(\)](#).

Referenced by [dbgprinttree\(\)](#).

#### 8.8.3.12 precedence()

```
unsigned GiNaC::basic::precedence () const [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [precedence\(\)](#).

Referenced by [precedence\(\)](#).

### 8.8.3.13 info()

```
bool GiNaC::basic::info (
    unsigned int) const [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::minkmetric](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::spinmetric](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::tensdelta](#), [GiNaC::tensepsilon](#), and [GiNaC::tensmetric](#).

References [info\(\)](#).

Referenced by [info\(\)](#), [GiNaC::function::info\(\)](#), and [GiNaC::matrix::solve\(\)](#).

### 8.8.3.14 nops()

```
size_t GiNaC::basic::nops () const [virtual]
```

Number of operands/members.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::expairseq](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::matrix](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::user\\_defined\\_kernel](#).

References [nops\(\)](#).

Referenced by [calchash\(\)](#), [derivative\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [has\(\)](#), [map\(\)](#), [match\(\)](#), [nops\(\)](#), [normal\(\)](#), and [subs\(\)](#).

### 8.8.3.15 op()

```
ex GiNaC::basic::op (
    size_t i) const [virtual]
```

Return operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::matrix](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::user\\_defined\\_kernel](#).

References [ex](#), and [op\(\)](#).

Referenced by [calchash\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [has\(\)](#), [map\(\)](#), [match\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), and [subs\(\)](#).

**8.8.3.16 operator[]() [1/4]**

```
ex GiNaC::basic::operator[] (
    const ex & index) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), and [GiNaC::to\\_int\(\)](#).

Referenced by [operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

**8.8.3.17 operator[]() [2/4]**

```
ex GiNaC::basic::operator[] (
    size_t i) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [op\(\)](#), and [operator\[\]\(\)](#).

**8.8.3.18 let\_op()**

```
ex & GiNaC::basic::let_op (
    size_t i) [virtual]
```

Return modifiable operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::integral](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::matrix](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::user\\_defined\\_kernel](#).

References [ensure\\_if\\_modifiable\(\)](#), [ex](#), and [let\\_op\(\)](#).

Referenced by [let\\_op\(\)](#), [map\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), and [subs\(\)](#).

**8.8.3.19 operator[]() [3/4]**

```
ex & GiNaC::basic::operator[] (
    const ex & index) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [let\\_op\(\)](#), [operator\[\]\(\)](#), and [GiNaC::to\\_int\(\)](#).

**8.8.3.20 operator[]() [4/4]**

```
ex & GiNaC::basic::operator[] (
    size_t i) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [let\\_op\(\)](#), and [operator\[\]\(\)](#).



### 8.8.3.21 has()

```
bool GiNaC::basic::has (
    const ex & pattern,
    unsigned options = 0) const [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented in [GiNaC::Gt](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [has\(\)](#), [match\(\)](#), [nops\(\)](#), and [op\(\)](#).

Referenced by [has\(\)](#), [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), [is\\_polynomial\(\)](#), and [series\(\)](#).

### 8.8.3.22 match()

```
bool GiNaC::basic::match (
    const ex & pattern,
    exmap & repl_lst) const [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::Gt](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::wildcard](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [is\\_equal\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [match\(\)](#), [match\\_same\\_type\(\)](#), [nops\(\)](#), [GiNaC::ex::nops\(\)](#), [op\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [has\(\)](#), [match\(\)](#), and [subs\\_one\\_level\(\)](#).

### 8.8.3.23 match\_same\_type()

```
bool GiNaC::basic::match_same_type (
    const basic & other) const [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::matrix](#), [GiNaC::relational](#), [GiNaC::spinidx](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::varidx](#).

References [basic\(\)](#), and [match\\_same\\_type\(\)](#).

Referenced by [match\(\)](#), and [match\\_same\\_type\(\)](#).

### 8.8.3.24 subs()

```
ex GiNaC::basic::subs (
    const exmap & m,
    unsigned options = 0) const [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basic\(\)](#), [clearflag\(\)](#), [duplicate\(\)](#), [ex](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::status\\_flags::hash\\_calculated](#), [let\\_op\(\)](#), [nops\(\)](#), [op\(\)](#), [subs\(\)](#), [GiNaC::ex::subs\(\)](#), and [subs\\_one\\_level\(\)](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [subs\(\)](#).

### 8.8.3.25 map()

```
ex GiNaC::basic::map (
    map_function & f) const [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented in [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basic\(\)](#), [clearflag\(\)](#), [duplicate\(\)](#), [ex](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::status\\_flags::hash\\_calculated](#), [let\\_op\(\)](#), [map\(\)](#), [nops\(\)](#), and [op\(\)](#).

Referenced by [derivative\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [map\(\)](#), and [normal\(\)](#).

### 8.8.3.26 accept()

```
virtual void GiNaC::basic::accept (
    GiNaC::visitor & v) const [inline], [virtual]
```

### 8.8.3.27 is\_polynomial()

```
bool GiNaC::basic::is_polynomial (
    const ex & var) const [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::symbol](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [has\(\)](#), [is\\_equal\(\)](#), and [is\\_polynomial\(\)](#).

Referenced by [is\\_polynomial\(\)](#).

**8.8.3.28 degree()**

```
int GiNaC::basic::degree (
    const ex & s) const [virtual]
```

Return degree of highest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [degree\(\)](#), [ex](#), [GiNaC::ex\\_to\(\)](#), and [is\\_equal\(\)](#).

Referenced by [collect\(\)](#), and [degree\(\)](#).

**8.8.3.29 ldegree()**

```
int GiNaC::basic::ldegree (
    const ex & s) const [virtual]
```

Return degree of lowest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [is\\_equal\(\)](#), and [ldegree\(\)](#).

Referenced by [collect\(\)](#), and [ldegree\(\)](#).

**8.8.3.30 coeff()**

```
ex GiNaC::basic::coeff (
    const ex & s,
    int n = 1) const [virtual]
```

Return coefficient of degree n in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [ex](#), [GiNaC::ex\\_to\(\)](#), and [is\\_equal\(\)](#).

Referenced by [coeff\(\)](#), [collect\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [series\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

**8.8.3.31 expand()**

```
ex GiNaC::basic::expand (
    unsigned options = 0) const [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [map\(\)](#), [nops\(\)](#), and [setflag\(\)](#).

Referenced by [collect\(\)](#), [expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

### 8.8.3.32 collect()

```
ex GiNaC::basic::collect (
    const ex & s,
    bool distributed = false) const [virtual]
```

Sort expanded expression in terms of powers of some object(s).

#### Parameters

<i>s</i>	object(s) to sort in
----------	----------------------

<i>distributed</i>	recursive or distributed form (only used when s is a list)
--------------------	--

Reimplemented in [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [collect\(\)](#), [GiNaC::ex::collect\(\)](#), [degree\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::dynallocate\(\)](#), [ex](#), [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [GiNaC::is\\_a\(\)](#), [ldegree\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::pow\(\)](#).

Referenced by [collect\(\)](#).

### 8.8.3.33 derivative()

```
ex GiNaC::basic::derivative (
    const symbol & s) const [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), [derivative\(\)](#), [ex](#), [map\(\)](#), and [nops\(\)](#).

Referenced by [derivative\(\)](#), and [diff\(\)](#).

### 8.8.3.34 series()

```
ex GiNaC::basic::series (
    const relational & r,
    int order,
    unsigned options = 0) const [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented in [GiNaC::add](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::integration\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::mul](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::numeric::div\(\)](#), [ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [has\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::lgamma\\_series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), and [GiNaC::power::series\(\)](#).

### 8.8.3.35 normal()

```
ex GiNaC::basic::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex1](#), [GiNaC::dynallocate\(\)](#), [ex](#), [GiNaC::ex::info\(\)](#), [GiNaC::is\\_a\(\)](#), [map\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::solve\(\)](#), and [GiNaC::Gt::tauToFundamental\(\)](#).

### 8.8.3.36 to\_rational()

```
ex GiNaC::basic::to_rational (
    exmap & repl) const [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [ex](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

### 8.8.3.37 to\_polynomial()

```
ex GiNaC::basic::to_polynomial (
    exmap & repl) const [virtual]
```

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [ex](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**8.8.3.38 integer\_content()**

```
numeric GiNaC::basic::integer_content () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

**8.8.3.39 smod()**

```
ex GiNaC::basic::smod (
    const numeric & xi) const [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$xi$	modulus
------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#).

**8.8.3.40 max\_coefficient()**

```
numeric GiNaC::basic::max_coefficient () const [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

#### 8.8.3.41 `get_free_indices()`

```
exvector GiNaC::basic::get_free_indices () const [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented in [GiNaC::add](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [get\\_free\\_indices\(\)](#).

Referenced by [get\\_free\\_indices\(\)](#).

#### 8.8.3.42 `add_indexed()`

```
ex GiNaC::basic::add_indexed (
    const ex & self,
    const ex & other) const [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class `indexed` (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

##### Parameters

<i>self</i>	First indexed expression; its base object is <code>*this</code>
<i>other</i>	Second indexed expression

##### Returns

sum of `self` and `other`

##### See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [add\\_indexed\(\)](#), and [ex](#).

Referenced by [add\\_indexed\(\)](#).

#### 8.8.3.43 `scalar_mul_indexed()`

```
ex GiNaC::basic::scalar_mul_indexed (
    const ex & self,
    const numeric & other) const [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

##### Parameters

<i>self</i>	Indexed expression; its base object is <code>*this</code>
-------------	---



<i>other</i>	Numeric value
--------------	---------------

**Returns**

product of self and other

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [ex](#), and [scalar\\_mul\\_indexed\(\)](#).

Referenced by [scalar\\_mul\\_indexed\(\)](#).

**8.8.3.44 contract\_with()**

```
bool GiNaC::basic::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

**Parameters**

<i>self</i>	Pointer to first indexed expression; its base object is *this
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

**Returns**

true if the contraction was successful, false otherwise

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::cliffordunit](#), [GiNaC::diracgamma](#), [GiNaC::matrix](#), [GiNaC::spinmetric](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::su3d](#), [GiNaC::su3f](#), [GiNaC::su3t](#), [GiNaC::tensdelta](#), [GiNaC::tensepsilon](#), and [GiNaC::tensmetric](#).

References [contract\\_with\(\)](#).

Referenced by [contract\\_with\(\)](#).

#### 8.8.3.45 `return_type()`

```
unsigned GiNaC::basic::return_type () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::expairseq](#), [GiNaC::fail](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::minkmetric](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::su3d](#), [GiNaC::su3f](#), [GiNaC::tensdelta](#), [GiNaC::tensepsilon](#), [GiNaC::tensmetric](#), and [GiNaC::tensor](#).

References [GiNaC::return\\_types::commutative](#), and [return\\_type\(\)](#).

Referenced by [return\\_type\(\)](#).

#### 8.8.3.46 `return_type_tinfo()`

```
return_type_t GiNaC::basic::return_type_tinfo () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [return\\_type\\_tinfo\(\)](#), [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [return\\_type\\_tinfo\(\)](#).

#### 8.8.3.47 `conjugate()`

```
ex GiNaC::basic::conjugate () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::diracgamma5](#), [GiNaC::diracgammaL](#), [GiNaC::diracgammaR](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::realsymbol](#), [GiNaC::spinidx](#), and [GiNaC::symbol](#).

References [conjugate\(\)](#), and [ex](#).

Referenced by [conjugate\(\)](#).

#### 8.8.3.48 `real_part()`

```
ex GiNaC::basic::real_part () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::realsymbol](#), and [GiNaC::symbol](#).

References [ex](#), and [real\\_part\(\)](#).

Referenced by [real\\_part\(\)](#), [GiNaC::function::real\\_part\(\)](#), and [GiNaC::ncmul::real\\_part\(\)](#).

### 8.8.3.49 imag\_part()

```
ex GiNaC::basic::imag_part () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::realsymbol](#), and [GiNaC::symbol](#).

References [ex](#), and [imag\\_part\(\)](#).

Referenced by [imag\\_part\(\)](#), [GiNaC::function::imag\\_part\(\)](#), and [GiNaC::ncmul::imag\\_part\(\)](#).

### 8.8.3.50 compare\_same\_type()

```
int GiNaC::basic::compare_same_type (
    const basic & other) const [protected], [virtual]
```

Returns order relation between two objects of same type.

This needs to be implemented by each class. It may never return anything else than 0, signalling equality, or +1 and -1 signalling inequality and determining the canonical ordering. (Perl hackers will wonder why C++ doesn't feature the spaceship operator `<=>` for denoting just this.)

References [basic\(\)](#), [GiNaC::compare\\_pointers\(\)](#), and [compare\\_same\\_type\(\)](#).

Referenced by [compare\(\)](#), [compare\\_same\\_type\(\)](#), [GiNaC::symbol::derivative\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

### 8.8.3.51 is\_equal\_same\_type()

```
bool GiNaC::basic::is_equal_same_type (
    const basic & other) const [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented in [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::numeric](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [basic\(\)](#), [compare\\_same\\_type\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

Referenced by [is\\_equal\(\)](#), [is\\_equal\\_same\\_type\(\)](#), and [match\(\)](#).

### 8.8.3.52 calchash()

```
unsigned GiNaC::basic::calchash () const [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented in [GiNaC::constant](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::numeric](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), and [GiNaC::wildcard](#).

References [calchash\(\)](#), [GiNaC::status\\_flags::evaluated](#), [flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [hashvalue](#), [nops\(\)](#), [op\(\)](#), [GiNaC::rotate\\_left\(\)](#), and [setflag\(\)](#).

Referenced by [calchash\(\)](#), and [gethash\(\)](#).

### 8.8.3.53 print\_dispatch() [1/2]

```
template<class T>
void GiNaC::basic::print_dispatch (
    const print\_context & c,
    unsigned level) const [inline]
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [print\\_dispatch\(\)](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [print\(\)](#), [print\\_dispatch\(\)](#), and [print\\_dispatch\(\)](#).

### 8.8.3.54 print\_dispatch() [2/2]

```
void GiNaC::basic::print_dispatch (
    const registered\_class\_info & ri,
    const print\_context & c,
    unsigned level) const
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [GiNaC::print\\_context\\_options::get\\_id\(\)](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GiNaC::class\\_info< OPT >::options](#), [print\\_dispatch\(\)](#), and [GiNaC::structure< T, CP >::reg\\_info](#).

### 8.8.3.55 archive()

```
void GiNaC::basic::archive (
    archive_node & n) const [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::minkmetric](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::spinidx](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::tensepsilon](#), [GiNaC::varidx](#), and [GiNaC::wildcard](#).

References [GiNaC::archive\\_node::add\\_string\(\)](#), and [archive\(\)](#).

Referenced by [archive\(\)](#).

### 8.8.3.56 read\_archive()

```
void GiNaC::basic::read_archive (
    const archive_node & n,
    lst & syms) [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::container< std::list >](#), [GiNaC::container< std::vector >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::Gt](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::minkmetric](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::spinidx](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::tensepsilon](#), [GiNaC::varidx](#), and [GiNaC::wildcard](#).

References [read\\_archive\(\)](#).

Referenced by [read\\_archive\(\)](#).

### 8.8.3.57 subs\_one\_level()

```
ex GiNaC::basic::subs_one_level (
    const exmap & m,
    unsigned options) const
```

Helper function for [subs\(\)](#).

Does not recurse into subexpressions.

References [ex](#), [GiNaC::ex\\_to\(\)](#), [match\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), and [subs\\_one\\_level\(\)](#).

Referenced by [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::symbol::subs\(\)](#), and [subs\\_one\\_level\(\)](#).

### 8.8.3.58 diff()

```
ex GiNaC::basic::diff (
    const symbol & s,
    unsigned nth = 1) const
```

Default interface of nth derivative [ex::diff\(s, n\)](#).

It should be called instead of [::derivative\(s\)](#) for first derivatives and for nth derivatives it just recurses down.

#### Parameters

<i>s</i>	symbol to differentiate in
<i>nth</i>	order of differentiation

#### See also

[ex::diff](#)

References [derivative\(\)](#), [diff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::status\\_flags::evaluated](#), [ex](#), [flags](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::mul::derivative\(\)](#), and [diff\(\)](#).

### 8.8.3.59 compare()

```
int GiNaC::basic::compare (
    const basic & other) const
```

Compare objects syntactically to establish canonical ordering.

All compare functions return: -1 for \*this less than other, 0 equal, 1 greater.

References [basic\(\)](#), [compare\(\)](#), [compare\\_same\\_type\(\)](#), and [gethash\(\)](#).

Referenced by [compare\(\)](#).

### 8.8.3.60 is\_equal()

```
bool GiNaC::basic::is_equal (
    const basic & other) const
```

Test for syntactic equality.

This is only a quick test, meaning objects should be in the same domain. You might have to `.expand()`, `.normal()` objects first, depending on the domain of your computation, to get a more reliable answer.

See also

[is\\_equal\\_same\\_type](#)

References [basic\(\)](#), [gethash\(\)](#), [is\\_equal\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

Referenced by [coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [degree\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [is\\_equal\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), [match\(\)](#), and [GiNaC::wildcard::match\(\)](#).

### 8.8.3.61 hold()

```
const basic & GiNaC::basic::hold () const
```

Stop further evaluation.

See also

[basic::eval](#)

References [basic\(\)](#), [GiNaC::status\\_flags::evaluated](#), [hold\(\)](#), and [setflag\(\)](#).

Referenced by [GiNaC::abs\\_conjugate\(\)](#), [GiNaC::abs\\_eval\(\)](#), [GiNaC::abs\\_evalf\(\)](#), [GiNaC::abs\\_expand\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::abs\\_real\\_part\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acos\\_evalf\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::acosh\\_evalf\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_evalf\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::asinh\\_evalf\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_evalf\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_evalf\(\)](#), [GiNaC::binomial\\_conjugate\(\)](#), [GiNaC::binomial\\_eval\(\)](#), [GiNaC::binomial\\_evalf\(\)](#), [GiNaC::binomial\\_real\\_part\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cos\\_evalf\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::cosh\\_evalf\(\)](#), [GiNaC::csgn\\_power\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::add::eval\(\)](#), [eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::fderivative::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::Gt::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::numeric::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_evalf\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::factorial\\_conjugate\(\)](#), [GiNaC::factorial\\_eval\(\)](#), [GiNaC::factorial\\_evalf\(\)](#), [GiNaC::factorial\\_real\\_part\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#), [hold\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), [GiNaC::iterated\\_integral2\\_eval\(\)](#), [GiNaC::iterated\\_integral3\\_eval\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_evalf\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_evalf\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::Order\\_power\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_evalf\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sin\\_evalf\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sinh\\_evalf\(\)](#), [GiNaC::step\\_conjugate\(\)](#), [GiNaC::step\\_eval\(\)](#), [GiNaC::step\\_evalf\(\)](#), [GiNaC::step\\_real\\_part\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tan\\_evalf\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tanh\\_evalf\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_eval\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).





**8.8.3.65 ensure\_if\_modifiable()**

```
void GiNaC::basic::ensure_if_modifiable () const [protected]
```

Ensure the object may be modified without hurting others, throws if this is not the case.

References [clearflag\(\)](#), [ensure\\_if\\_modifiable\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

Referenced by [GiNaC::container< C >::append\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [ensure\\_if\\_modifiable\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [let\\_op\(\)](#), [GiNaC::clifford::let\\_op\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::Ebar\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_kernel::let\\_op\(\)](#), [GiNaC::ELi\\_kernel::let\\_op\(\)](#), [GiNaC::integral::let\\_op\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::let\\_op\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::let\\_op\(\)](#), [GiNaC::matrix::let\\_op\(\)](#), [GiNaC::modular\\_form\\_kernel::let\\_op\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::let\\_op\(\)](#), [GiNaC::user\\_defined\\_kernel::let\\_op\(\)](#), [GiNaC::matrix::operator\(\)\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::container< C >::prepend\(\)](#), [GiNaC::container< C >::remove\\_all\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), [GiNaC::container< C >::remove\\_last\(\)](#), [GiNaC::container< C >::sort\(\)](#), and [GiNaC::container< C >::unique\(\)](#).

**8.8.3.66 do\_print()**

```
void GiNaC::basic::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

Default output to stream.

References [do\\_print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\(\)](#), and [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.8.3.67 do\_print\_tree()**

```
void GiNaC::basic::do_print_tree (
    const print_tree & c,
    unsigned level) const [protected]
```

Tree output to stream.

References [GiNaC::print\\_tree::delta\\_indent](#), [do\\_print\\_tree\(\)](#), [flags](#), [hashvalue](#), [nops\(\)](#), [op\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\\_tree\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), and [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.8.3.68 do\_print\_python\_repr()**

```
void GiNaC::basic::do_print_python_repr (
    const print_python_repr & c,
    unsigned level) const [protected]
```

Python parsable output to stream.

References [do\\_print\\_python\\_repr\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\\_python\\_repr\(\)](#), and [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.8.4 Friends And Related Symbol Documentation

### 8.8.4.1 ex

friend class ex [friend]

References [ex](#).

Referenced by [add\\_indexed\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::add\\_indexed\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::container< std::vector >::append\(\)](#), [GiNaC::Gt::apply\\_function\\_recursive\(\)](#), [GiNaC::Gt::applyIntegrationPath\(\)](#), [GiNaC::relational::canonical\(\)](#), [GiNaC::matrix::charpoly\(\)](#), [coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::coeff\(\)](#), [GiNaC::pseries::coeffop\(\)](#), [collect\(\)](#), [GiNaC::pseries::collect\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::collect\(\)](#), [GiNaC::expairseq::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [GiNaC::constant::conjugate\(\)](#), [GiNaC::container< std::vector >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::numeric::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::symbol::conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::expairseq::default\\_overall\\_coeff\(\)](#), [degree\(\)](#), [GiNaC::integral::degree\(\)](#), [GiNaC::numeric::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::pseries::degree\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::degree\(\)](#), [derivative\(\)](#), [GiNaC::constant::derivative\(\)](#), [GiNaC::idx::derivative\(\)](#), [GiNaC::integral::derivative\(\)](#), [GiNaC::numeric::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [diff\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::Gt::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::numeric::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\(\)](#), [GiNaC::symbol::eval\(\)](#), [eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\\_indexed\(\)](#), [eval\\_integ\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [GiNaC::integral::eval\\_ncmul\(\)](#), [GiNaC::power::eval\\_ncmul\(\)](#), [GiNaC::relational::eval\\_ncmul\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\\_ncmul\(\)](#), [evalf\(\)](#), [GiNaC::constant::evalf\(\)](#), [GiNaC::Gt::evalf\(\)](#), [GiNaC::idx::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::numeric::evalf\(\)](#), [GiNaC::power::evalf\(\)](#), [GiNaC::pseries::evalf\(\)](#), [GiNaC::symbol::evalf\(\)](#), [evalm\(\)](#), [GiNaC::matrix::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::evalm\(\)](#), [GiNaC::Gt::evaluate\(\)](#), [ex](#), [GiNaC::Gt::ex\\_cutIntegrationPath\(\)](#), [GiNaC::Gt::ex\\_cutIntegrationPath\(\)](#), [GiNaC::Gt::ex\\_evaluate\(\)](#), [GiNaC::Gt::ex\\_evaluate\(\)](#), [GiNaC::Gt::ex\\_prepare\(\)](#), [GiNaC::Gt::ex\\_prepare\(\)](#), [GiNaC::Gt::ex\\_qExpand\(\)](#), [GiNaC::Gt::ex\\_qExpand\(\)](#), [GiNaC::Gt::ex\\_regularise\(\)](#), [GiNaC::Gt::ex\\_regularise\(\)](#), [GiNaC::Gt::ex\\_tauToFundamental\(\)](#), [GiNaC::Gt::ex\\_tauToFundamental\(\)](#), [GiNaC::Gt::ex\\_zisToFundamental\(\)](#), [GiNaC::Gt::ex\\_zisToFundamental\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::pseries::exponop\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::pseries::get\\_point\(\)](#), [GiNaC::integration\\_kernel::get\\_series\\_coeff\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GiNaC::pseries::get\\_var\(\)](#), [GiNaC::Gt::getTau\(\)](#), [GiNaC::Gt::getZ\(\)](#), [GiNaC::Gt::Gt\(\)](#), [GiNaC::Gt::Gt\(\)](#), [has\(\)](#), [GiNaC::Gt::has\(\)](#), [GiNaC::numeric::has\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::has\(\)](#), [GiNaC::idx::idx\(\)](#), [imag\\_part\(\)](#), [GiNaC::constant::imag\\_part\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::matrix::imag\\_part\(\)](#), [GiNaC::numeric::imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [GiNaC::symbol::imag\\_part\(\)](#), [GiNaC::integral::integral\(\)](#), [is\\_polynomial\(\)](#), [GiNaC::constant::is\\_polynomial\(\)](#), [GiNaC::numeric::is\\_polynomial\(\)](#), [GiNaC::power::is\\_polynomial\(\)](#), [GiNaC::symbol::is\\_polynomial\(\)](#), [GiNaC::Gt::kernel::kernel\(\)](#), [GiNaC::Gt::kernel::kernel\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [lddegree\(\)](#), [GiNaC::integral::lddegree\(\)](#), [GiNaC::numeric::lddegree\(\)](#), [GiNaC::power::lddegree\(\)](#), [GiNaC::pseries::lddegree\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::lddegree\(\)](#), [let\\_op\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::container< std::vector >::let\\_op\(\)](#), [GiNaC::integral::let\\_op\(\)](#), [GiNaC::matrix::let\\_op\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::let\\_op\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [map\(\)](#), [GiNaC::expairseq::map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::power::map\(\)](#), [GiNaC::relational::map\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::map\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [match\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::Gt::match\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::match\(\)](#), [GiNaC::wildcard::match\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [normal\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::normal\(\)](#), [GiNaC::symbol::normal\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::idx::op\(\)](#), [GiNaC::integral::op\(\)](#), [GiNaC::matrix::op\(\)](#),

[GiNaC::power::op\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::relational::op\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::op\(\)](#),  
[GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::matrix::operator\(\)\(\)](#), [GiNaC::matrix::operator\(\)\(\)](#), [operator\[\]\(\)](#),  
[operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::operator\[\]\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::operator\[\]\(\)](#),  
[GiNaC::matrix::pow\(\)](#), [GiNaC::power::power\(\)](#), [GiNaC::power::power\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::container< C >::prepend\(\)](#),  
[GiNaC::container< std::vector >::prepend\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::Gt::qExpand\(\)](#),  
[GiNaC::container< std::vector >::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::Gt::read\\_archive\(\)](#),  
[GiNaC::matrix::read\\_archive\(\)](#), [GiNaC::pseries::read\\_archive\(\)](#), [GiNaC::symmetry::read\\_archive\(\)](#), [real\\_part\(\)](#),  
[GiNaC::constant::real\\_part\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), [GiNaC::matrix::real\\_part\(\)](#), [GiNaC::numeric::real\\_part\(\)](#),  
[GiNaC::power::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::symbol::real\\_part\(\)](#), [GiNaC::expairseq::recombine\\_pair\\_to\\_ex\(\)](#),  
[GiNaC::Gt::regularise\(\)](#), [GiNaC::relational::relational\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#),  
[GiNaC::relational::rhs\(\)](#), [scalar\\_mul\\_indexed\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::scalar\\_mul\\_indexed\(\)](#),  
[series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#),  
[GiNaC::structure< T, ComparisonPolicy >::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#),  
[GiNaC::matrix::set\(\)](#), [GiNaC::Gt::setArgs\(\)](#), [smod\(\)](#), [GiNaC::numeric::smod\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::smod\(\)](#),  
[GiNaC::matrix::solve\(\)](#), [GiNaC::expairseq::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::container< std::vector >::subs\(\)](#),  
[GiNaC::expairseq::subs\(\)](#), [GiNaC::Gt::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::numeric::subs\(\)](#),  
[GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::subs\(\)](#),  
[GiNaC::symbol::subs\(\)](#), [subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subchildren\(\)](#), [GiNaC::expairseq::subchildren\(\)](#),  
[GiNaC::Gt::tauToFundamental\(\)](#), [GiNaC::container< std::vector >::thiscontainer\(\)](#), [GiNaC::container< std::vector >::thiscontainer\(\)](#),  
[GiNaC::expairseq::thisexpairseq\(\)](#), [GiNaC::expairseq::thisexpairseq\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), [GiNaC::Gt::to\\_numeric\(\)](#),  
[to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::numeric::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#),  
[GiNaC::structure< T, ComparisonPolicy >::to\\_polynomial\(\)](#), [GiNaC::symbol::to\\_polynomial\(\)](#), [to\\_rational\(\)](#),  
[GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::numeric::to\\_rational\(\)](#), [GiNaC::power::to\\_rational\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::to\\_rational\(\)](#),  
[GiNaC::symbol::to\\_rational\(\)](#), [GiNaC::matrix::trace\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

## 8.8.5 Member Data Documentation

### 8.8.5.1 flags

`unsigned GiNaC::basic::flags [mutable], [protected]`

of type [status\\_flags](#)

Referenced by [basic\(\)](#), [calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::Gt::calchash\(\)](#),  
[GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [clearflag\(\)](#), [GiNaC::ex::construct\\_from\\_basic\(\)](#),  
[diff\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tree\(\)](#),  
[GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#),  
[GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#),  
[GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#),  
[GiNaC::expairseq::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::Gt::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#),  
[GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [gethash\(\)](#),  
[GiNaC::expairseq::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [operator=\(\)](#), [GiNaC::function::print\(\)](#), [setflag\(\)](#),  
 and [~basic\(\)](#).

### 8.8.5.2 hashvalue

`unsigned GiNaC::basic::hashvalue [mutable], [protected]`

hash value

Referenced by [calchash\(\)](#), [GiNaC::constant::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#),  
[GiNaC::Gt::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#),  
[GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#),

[GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [gethash\(\)](#), [operator=\(\)](#), and [GiNaC::function::print\(\)](#).

The documentation for this class was generated from the following files:

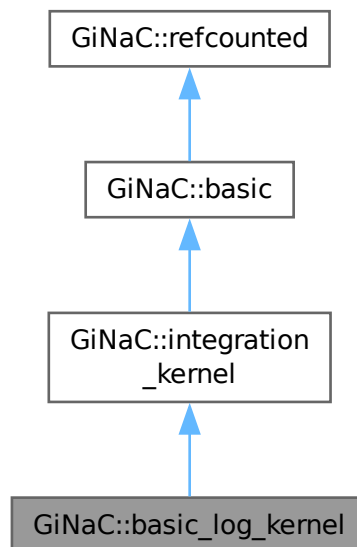
- [basic.h](#)
- [basic.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 8.9 GiNaC::basic\_log\_kernel Class Reference

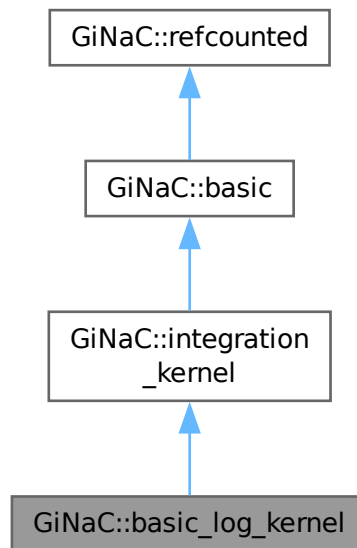
The basic integration kernel with a logarithmic singularity at the origin.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::basic\_log\_kernel:



Collaboration diagram for GiNaC::basic\_log\_kernel:



### Protected Member Functions

- `cln::cl_N_series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*

- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int i) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*

- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprntree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept



## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.9.1 Detailed Description

The basic integration kernel with a logarithmic singularity at the origin.

This class represents the differential one-form

$$L_0 = \frac{d\lambda}{\lambda}$$

### 8.9.2 Member Function Documentation

#### 8.9.2.1 [series\\_coeff\\_impl\(\)](#)

```
cln::cl_N GiNaC::basic_log_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [series\\_coeff\\_impl\(\)](#).

Referenced by [series\\_coeff\\_impl\(\)](#).

#### 8.9.2.2 [do\\_print\(\)](#)

```
void GiNaC::basic_log_kernel::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [do\\_print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\(\)](#), and [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

The documentation for this class was generated from the following files:

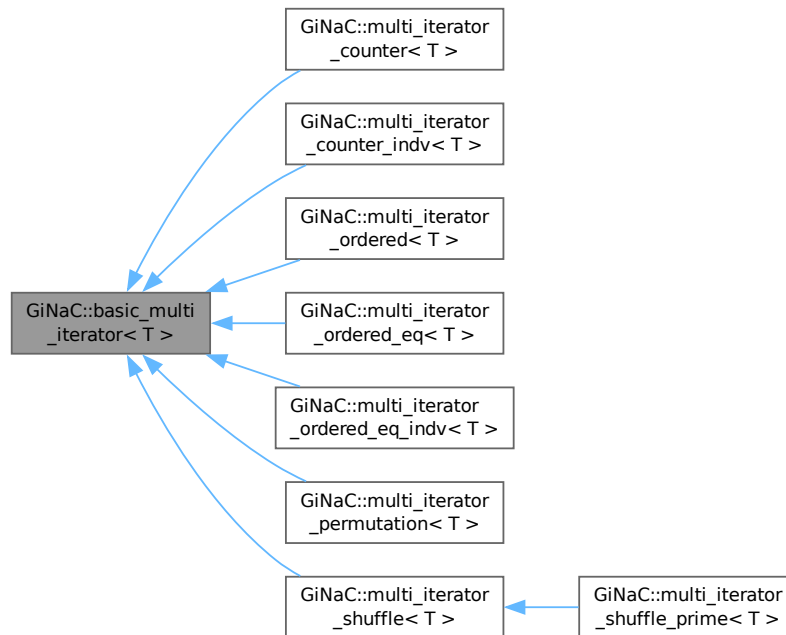
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.10 GiNaC::basic\_multi\_iterator< T > Class Template Reference

[basic\\_multi\\_iterator](#) is a base class.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::basic\_multi\_iterator< T >:



### Public Member Functions

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)

- Subscription via [].*
- T [operator\(\)](#) (size\_t i) const
- Subscription via ().*
- T & [operator\(\)](#) (size\_t i)
- Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)
- Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)
- No effect for [basic\\_multi\\_iterator](#).*

### Protected Attributes

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

### Friends

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [basic\\_multi\\_iterator](#)< TT > &v)

## 8.10.1 Detailed Description

**template<class T>**  
**class GiNaC::basic\_multi\_iterator< T >**

[basic\\_multi\\_iterator](#) is a base class.

The base class itself does not do anything useful. A typical use of a class derived from [basic\\_multi\\_iterator](#) is

```
multi\_iterator\_ordered<int> k(0,4,2);
```

```
for( k.init(); !k.overflow\(\); k++) { std::cout << k << std::endl; }
```

which prints out

```
multi_iterator_ordered(0,1) multi_iterator_ordered(0,2) multi_iterator_ordered(0,3) multi_iterator_ordered(1,2)
multi_iterator_ordered(1,3) multi_iterator_ordered(2,3)
```

Individual components of k can be accessed with k[i] or k(i).

All classes derived from [basic\\_multi\\_iterator](#) follow the same syntax.

## 8.10.2 Constructor & Destructor Documentation

### 8.10.2.1 `basic_multi_iterator()` [1/3]

```
template<class T>
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    void )    [inline]
```

Default constructor.

References [B](#), [flag\\_overflow](#), [N](#), and [v](#).

Referenced by `init()`, `GiNaC::multi_iterator_counter< T >::init()`, `GiNaC::multi_iterator_counter_indv< T >::init()`, `GiNaC::multi_iterator_ordered< T >::init()`, `GiNaC::multi_iterator_ordered_eq< T >::init()`, `GiNaC::multi_iterator_ordered_eq_indv< T >::init()`, `GiNaC::multi_iterator_permutation< T >::init()`, `GiNaC::multi_iterator_shuffle< T >::init()`, `GiNaC::multi_iterator_counter< T >::multi_iterator_counter()`, `GiNaC::multi_iterator_counter< T >::multi_iterator_counter_indv()`, `GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv()`, `GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered()`, `GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq()`, `GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv()`, `GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation()`, `GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle()`, `GiNaC::multi_iterator_counter< T >::operator++()`, `GiNaC::multi_iterator_counter_indv< T >::operator++()`, `GiNaC::multi_iterator_ordered< T >::operator++()`, `GiNaC::multi_iterator_ordered_eq< T >::operator++()`, `GiNaC::multi_iterator_ordered_eq_indv< T >::operator++()`, `GiNaC::multi_iterator_permutation< T >::operator++()`, `GiNaC::multi_iterator_shuffle< T >::operator++()`, and `operator<<`.

### 8.10.2.2 `basic_multi_iterator()` [2/3]

```
template<class T>
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    size_t k)    [inline], [explicit]
```

Construct a `multi_iterator` with upper limit `N`, lower limit `B` and size `k`.

References [B](#), [flag\\_overflow](#), [N](#), and [v](#).

### 8.10.2.3 `basic_multi_iterator()` [3/3]

```
template<class T>
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    const std::vector< T > & vv)    [inline], [explicit]
```

Construct from a vector.

References [B](#), [flag\\_overflow](#), [N](#), and [v](#).

#### 8.10.2.4 ~basic\_multi\_iterator()

```
template<class T>
GiNaC::basic_multi_iterator< T >::~~basic_multi_iterator () [inline], [virtual]
```

Destructor.

### 8.10.3 Member Function Documentation

#### 8.10.3.1 size()

```
template<class T>
size_t GiNaC::basic_multi_iterator< T >::size (
    void ) const [inline]
```

Returns the size of a multi\_iterator.

References [v](#).

Referenced by [GiNaC::multi\\_iterator\\_permutation< T >::get\\_sign\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), and [GiNaC::operator<<\(\)](#).

#### 8.10.3.2 overflow()

```
template<class T>
bool GiNaC::basic_multi_iterator< T >::overflow (
    void ) const [inline]
```

Return the overflow flag.

References [flag\\_overflow](#).

Referenced by [GiNaC::Gt::regularise\(\)](#).

#### 8.10.3.3 get\_vector()

```
template<class T>
const std::vector< T > & GiNaC::basic_multi_iterator< T >::get_vector (
    void ) const [inline]
```

Returns a reference to the vector v.

References [v](#).

Referenced by [GiNaC::Gt::regularise\(\)](#).

**8.10.3.4 operator[]() [1/2]**

```
template<class T>
T GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i) const [inline]
```

Subscription via [].

References [v](#).

**8.10.3.5 operator[]() [2/2]**

```
template<class T>
T & GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i) [inline]
```

Subscription via [].

References [v](#).

**8.10.3.6 operator()() [1/2]**

```
template<class T>
T GiNaC::basic_multi_iterator< T >::operator() (
    size_t i) const [inline]
```

Subscription via ().

References [v](#).

**8.10.3.7 operator()() [2/2]**

```
template<class T>
T & GiNaC::basic_multi_iterator< T >::operator() (
    size_t i) [inline]
```

Subscription via ().

References [v](#).

**8.10.3.8 init()**

```
template<class T>
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented in [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), [GiNaC::multi\\_iterator\\_shuffle< T >](#), and [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

References [B](#), [basic\\_multi\\_iterator\(\)](#), [flag\\_overflow](#), and [v](#).

### 8.10.3.9 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::operator++ (
    int ) [inline], [virtual]
```

No effect for [basic\\_multi\\_iterator](#).

Reimplemented in [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), and [GiNaC::multi\\_iterator\\_shuffle< T >](#).

References [basic\\_multi\\_iterator\(\)](#).

## 8.10.4 Friends And Related Symbol Documentation

### 8.10.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const basic_multi_iterator< TT > & v) [friend]
```

References [basic\\_multi\\_iterator\(\)](#), and [v](#).

## 8.10.5 Member Data Documentation

### 8.10.5.1 N

```
template<class T>
T GiNaC::basic_multi_iterator< T >::N [protected]
```

Referenced by [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::multi\\_iterator\\_counter\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::multi\\_iterator\\_counter\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::multi\\_iterator\\_ordered\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::multi\\_iterator\\_ordered\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::multi\\_iterator\\_ordered\\_eq\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::multi\\_iterator\\_ordered\\_eq\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::multi\\_iterator\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::multi\\_iterator\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), and [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#).

### 8.10.5.2 B

```
template<class T>
T GiNaC::basic_multi_iterator< T >::B [protected]
```

Referenced by [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::multi\\_iterator\\_counter\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::multi\\_iterator\\_counter\\_indv\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::multi\\_iterator\\_ordered\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::multi\\_iterator\\_ordered\\_eq\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::multi\\_iterator\\_ordered\\_eq\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::multi\\_iterator\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), and [GiNaC::operator<<\(\)](#).

### 8.10.5.3 v

```
template<class T>
std::vector<T> GiNaC::basic_multi_iterator< T >::v [protected]
```

Referenced by [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::get\\_sign\(\)](#), [get\\_vector\(\)](#), [init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::multi\\_iterator\\_counter\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::multi\\_iterator\\_counter\\_indv\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::multi\\_iterator\\_ordered\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::multi\\_iterator\\_ordered\\_eq\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::multi\\_iterator\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#), [operator\(\)](#), [operator\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::operator++\(\)](#), [operator<<\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::operator<<\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::operator<<\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), and [size\(\)](#).

### 8.10.5.4 flag\_overflow

```
template<class T>
bool GiNaC::basic_multi_iterator< T >::flag_overflow [protected]
```

Referenced by [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [basic\\_multi\\_iterator\(\)](#), [init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::init\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::operator++\(\)](#), and [overflow\(\)](#).

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

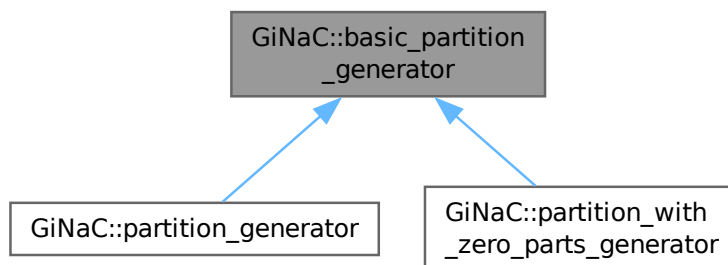
## 8.11 GiNaC::basic\_partition\_generator Class Reference

Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

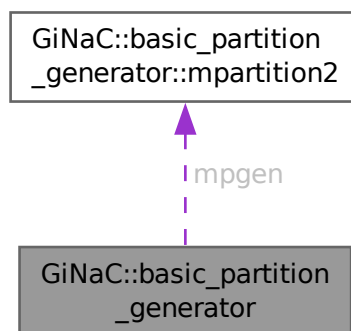
```
#include <utils.h>
```



Inheritance diagram for GiNaC::basic\_partition\_generator:



Collaboration diagram for GiNaC::basic\_partition\_generator:



## Classes

- struct `mpartition2`

## Protected Member Functions

- `basic_partition_generator` (unsigned `n_`, unsigned `m_`)

## Protected Attributes

- `mpartition2` `mpgen`

### 8.11.1 Detailed Description

Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

### 8.11.2 Constructor & Destructor Documentation

#### 8.11.2.1 `basic_partition_generator()`

```
GiNaC::basic_partition_generator::basic_partition_generator (
    unsigned n_,
    unsigned m_) [inline], [protected]
```

References [mpgen](#).

Referenced by [GiNaC::partition\\_generator::partition\\_generator\(\)](#), and [GiNaC::partition\\_with\\_zero\\_parts\\_generator::partition\\_with\\_zero\\_parts\\_generator\(\)](#).

### 8.11.3 Member Data Documentation

#### 8.11.3.1 `mpgen`

```
mpartition2 GiNaC::basic_partition_generator::mpgen [protected]
```

Referenced by [basic\\_partition\\_generator\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::next\(\)](#), and [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#).

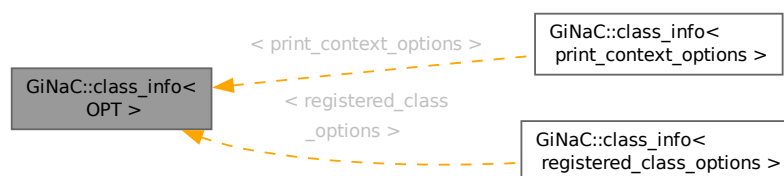
The documentation for this class was generated from the following file:

- [utils.h](#)

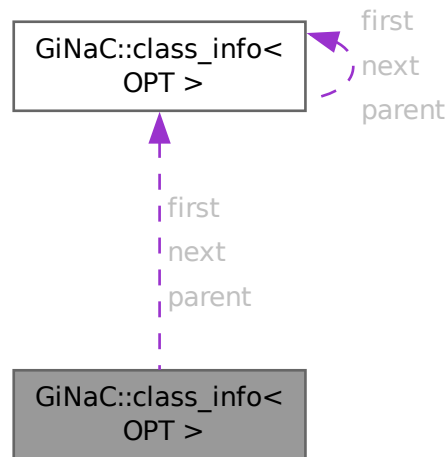
## 8.12 GiNaC::class\_info< OPT > Class Template Reference

```
#include <class_info.h>
```

Inheritance diagram for GiNaC::class\_info< OPT >:



Collaboration diagram for GiNaC::class\_info< OPT >:



## Classes

- struct [tree\\_node](#)

## Public Member Functions

- [class\\_info](#) (const OPT &o)
- [class\\_info](#) \* [get\\_parent](#) () const  
*Get pointer to [class\\_info](#) of parent class (or nullptr).*

## Static Public Member Functions

- static const [class\\_info](#) \* [find](#) (const std::string &class\_name)  
*Find [class\\_info](#) by name.*
- static void [dump\\_hierarchy](#) (bool verbose=false)  
*Dump class hierarchy to std::cout.*

## Public Attributes

- OPT [options](#)

## Static Private Member Functions

- static void [dump\\_tree](#) ([tree\\_node](#) \*n, const std::string &prefix, bool verbose)
- static void [identify\\_parents](#) ()

### Private Attributes

- [class\\_info](#) \* [next](#)
- [class\\_info](#) \* [parent](#)

### Static Private Attributes

- static [class\\_info](#) \* [first](#)
- static bool [parents\\_identified](#)

## 8.12.1 Constructor & Destructor Documentation

### 8.12.1.1 [class\\_info\(\)](#)

```
template<class OPT>
GiNaC::class_info< OPT >::class_info (
    const OPT & o) [inline]
```

Referenced by [dump\\_hierarchy\(\)](#), [find\(\)](#), [identify\\_parents\(\)](#), and [GiNaC::class\\_info< OPT >::tree\\_node::tree\\_node\(\)](#).

## 8.12.2 Member Function Documentation

### 8.12.2.1 [get\\_parent\(\)](#)

```
template<class OPT>
class_info * GiNaC::class_info< OPT >::get_parent () const [inline]
```

Get pointer to [class\\_info](#) of parent class (or nullptr).

Referenced by [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

### 8.12.2.2 [find\(\)](#)

```
template<class OPT>
const class_info< OPT > * GiNaC::class_info< OPT >::find (
    const std::string & class_name) [static]
```

Find [class\\_info](#) by name.

References [class\\_info\(\)](#), [find\(\)](#), [first](#), [next](#), and [options](#).

Referenced by [find\(\)](#).

### 8.12.2.3 [dump\\_hierarchy\(\)](#)

```
template<class OPT>
void GiNaC::class_info< OPT >::dump_hierarchy (
    bool verbose = false) [static]
```

Dump class hierarchy to `std::cout`.

References [class\\_info\(\)](#), [dump\\_tree\(\)](#), [first](#), [identify\\_parents\(\)](#), and [GiNaC::tree\(\)](#).

### 8.12.2.4 dump\_tree()

```
template<class OPT>
void GiNaC::class_info< OPT >::dump_tree (
    tree_node * n,
    const std::string & prefix,
    bool verbose) [static], [private]
```

References [GiNaC::class\\_info< OPT >::tree\\_node::children](#), [dump\\_tree\(\)](#), [GiNaC::class\\_info< OPT >::tree\\_node::info](#), and [options](#).

Referenced by [dump\\_hierarchy\(\)](#), and [dump\\_tree\(\)](#).

### 8.12.2.5 identify\_parents()

```
template<class OPT>
void GiNaC::class_info< OPT >::identify_parents () [static], [private]
```

References [class\\_info\(\)](#), [first](#), and [parents\\_identified](#).

Referenced by [dump\\_hierarchy\(\)](#), and [GiNaC::class\\_info< print\\_context\\_options >::get\\_parent\(\)](#).

## 8.12.3 Member Data Documentation

### 8.12.3.1 options

```
template<class OPT>
OPT GiNaC::class_info< OPT >::options
```

Referenced by [dump\\_tree\(\)](#), [find\(\)](#), [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

### 8.12.3.2 first

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::first [static], [private]
```

Referenced by [dump\\_hierarchy\(\)](#), [find\(\)](#), and [identify\\_parents\(\)](#).

### 8.12.3.3 next

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::next [private]
```

Referenced by [find\(\)](#).

### 8.12.3.4 parent

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::parent [mutable], [private]
```

### 8.12.3.5 parents\_identified

```
template<class OPT>
bool GiNaC::class_info< OPT >::parents_identified [static], [private]
```

Referenced by [identify\\_parents\(\)](#).

The documentation for this class was generated from the following file:

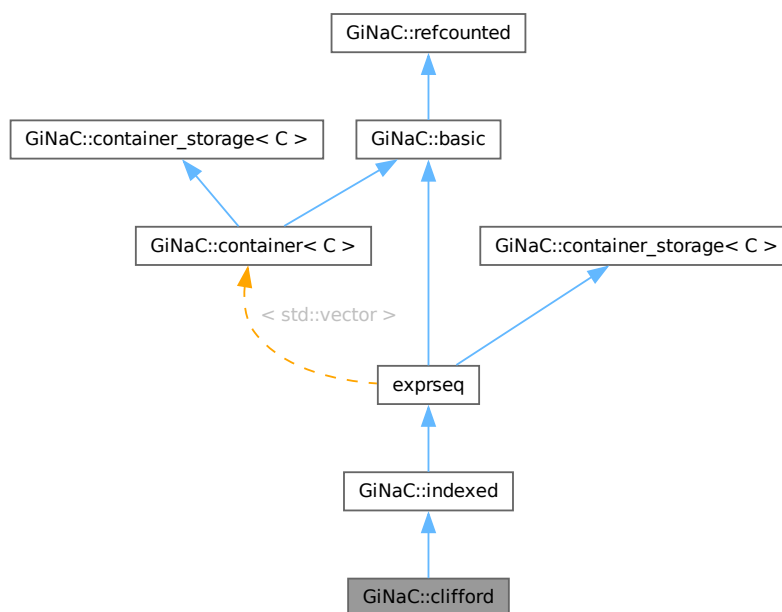
- [class\\_info.h](#)

## 8.13 GiNaC::clifford Class Reference

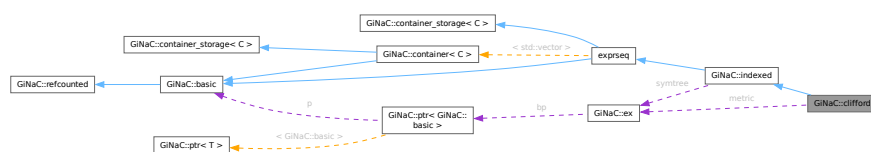
This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::clifford:



Collaboration diagram for GiNaC::clifford:



## Public Member Functions

- `clifford` (const `ex` &b, unsigned char rl=0)  
*Construct object without any indices.*
- `clifford` (const `ex` &b, const `ex` &mu, const `ex` &metr, unsigned char rl=0, int comm\_sign=-1)  
*Construct object with one Lorentz index.*
- `clifford` (unsigned char rl, const `ex` &metr, int comm\_sign, const `exvector` &v)
- `clifford` (unsigned char rl, const `ex` &metr, int comm\_sign, `exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char `get_representation_label` () const
- `ex` `get_metric` () const
- virtual `ex` `get_metric` (const `ex` &i, const `ex` &j, bool symmetrised=false) const
- bool `same_metric` (const `ex` &other) const
- int `get_commutator_sign` () const
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex` `op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex` & `let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex` `subs` (const `exmap` &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*

Public Member Functions inherited from `GiNaC::indexed`

- `indexed` (const `ex` &b)  
*Construct indexed object with no index.*
- `indexed` (const `ex` &b, const `ex` &i1)  
*Construct indexed object with one index.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
*Construct indexed object with three indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
*Construct indexed object with four indices.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
*Construct indexed object with three indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
*Construct indexed object with four indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `exvector` &iv)  
*Construct indexed object with a specified vector of indices.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)  
*Construct indexed object with a specified vector of indices and symmetry.*
- `indexed` (const `symmetry` &symm, const `exprseq` &es)

- `indexed` (const `symmetry` &`symm`, const `exvector` &`v`)
- `indexed` (const `symmetry` &`symm`, `exvector` &&`v`)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- void `archive` (`archive_node` &`n`) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
*Read (a.k.a.*
- bool `all_index_values_are` (unsigned inf) const  
*Check whether all index values have a certain property.*
- `exvector get_indices` () const  
*Return a vector containing the object's indices.*
- `exvector get_dummy_indices` () const  
*Return a vector containing the dummy indices of the object, if any.*
- `exvector get_dummy_indices` (const `indexed` &`other`) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- bool `has_dummy_index_for` (const `ex` &`i`) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- `ex get_symmetry` () const  
*Return symmetry properties.*

## Public Member Functions inherited from `GiNaC::container< std::vector >`

- `container` (STLT const &`s`)
- `container` (STLT &&`v`)
- `container` (`exvector::const_iterator` `b`, `exvector::const_iterator` `e`)
- `container` (`std::initializer_list< ex >` `il`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (`size_t` `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (`size_t` `i`) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &`m`, unsigned `options=0`) const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend` (const `ex` &`b`)  
*Add element at front.*
- `container & append` (const `ex` &`b`)  
*Add element at back.*
- `container & remove_first` ()  
*Remove first element.*
- `container & remove_last` ()



- Remove last element.*
- `container & remove_all ()`
- Remove all elements.*
- `container & sort ()`
- Sort elements.*
- `container & unique ()`
- Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`

- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- template<class T>
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const

*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override  
*Perform automatic simplification on noncommutative product of clifford objects.*
- bool `match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &`v`) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&`v`) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- void `do_print_dflt` (const `print_dflt` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const

### Protected Member Functions inherited from `GiNaC::indexed`

- `ex derivative` (const `symbol` &`s`) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &`v`) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&`v`) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &`c`, unsigned level) const
- void `print_indexed` (const `print_context` &`c`, const char \*`openbrace`, const char \*`closebrace`, unsigned level) const
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

### Protected Member Functions inherited from `GiNaC::container< std::vector >`

- `ex conjugate` () const override
- bool `is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- virtual void `printseq` (const `print_context` &`c`, char `openbracket`, char `delim`, char `closebracket`, unsigned `this_precedence`, unsigned `upper_precedence`=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const
- void `do_print_python` (const `print_python` &`c`, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const
- `STLT subchildren` (const `exmap` &`m`, unsigned options=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent spin lines.*
- [ex metric](#)  
*Metric of the space, all constructors make it an indexed object.*
- int [commutator\\_sign](#)  
*It is the sign in the definition  $e \sim i e \sim j +/ - e \sim j e \sim i = B(i, j) + B(j, i)$ .*

## Protected Attributes inherited from [GiNaC::indexed](#)

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects).*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**Additional Inherited Members****Public Types inherited from [GiNaC::container< std::vector >](#)**

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

**Protected Types inherited from [GiNaC::container< std::vector >](#)**

- typedef [container\\_storage< std::vector >::STLT](#) [STLT](#)

**Protected Types inherited from [GiNaC::container\\_storage< C >](#)**

- typedef C< [ex](#) > [STLT](#)

**Static Protected Member Functions inherited from [GiNaC::container< std::vector >](#)**

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) ([STLT](#) &, size\_t)

**8.13.1 Detailed Description**

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

These objects only carry Lorentz indices. Spinor indices are hidden. A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Clifford algebras (objects with different labels commute).

**8.13.2 Constructor & Destructor Documentation****8.13.2.1 [clifford\(\)](#) [1/4]**

```
GiNaC::clifford::clifford (
    const ex & b,
    unsigned char r1 = 0)
```

Construct object without any indices.

This constructor is for internal use only. Use the [dirac\\_ONE\(\)](#) function instead.

See also

[dirac\\_ONE](#)

References [clifford\(\)](#), [commutator\\_sign](#), [metric](#), and [representation\\_label](#).

Referenced by [clifford\(\)](#), [eval\\_ncmul\(\)](#), [match\\_same\\_type\(\)](#), [subs\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

### 8.13.2.2 `clifford()` [2/4]

```
GiNaC::clifford::clifford (
    const ex & b,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0,
    int comm_sign = -1)
```

Construct object with one Lorentz index.

This constructor is for internal use only. Use the [clifford\\_unit\(\)](#) or [dirac\\_gamma\(\)](#) functions instead.

See also

[clifford\\_unit](#)

[dirac\\_gamma](#)

References [commutator\\_sign](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [metric](#), and [representation\\_label](#).

### 8.13.2.3 `clifford()` [3/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    const exvector & v)
```

References [commutator\\_sign](#), [metric](#), [GiNaC::not\\_symmetric\(\)](#), and [representation\\_label](#).

### 8.13.2.4 `clifford()` [4/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    exvector && v)
```

References [commutator\\_sign](#), [metric](#), [GiNaC::not\\_symmetric\(\)](#), and [representation\\_label](#).

## 8.13.3 Member Function Documentation

### 8.13.3.1 `precedence()`

```
unsigned GiNaC::clifford::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_dflt\(\)](#), and [do\\_print\\_latex\(\)](#).

### 8.13.3.2 archive()

```
void GiNaC::clifford::archive (
    archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Loosely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [commutator\\_sign](#), [metric](#), and [representation\\_label](#).

### 8.13.3.3 read\_archive()

```
void GiNaC::clifford::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [metric](#), and [representation\\_label](#).

### 8.13.3.4 eval\_ncmul()

```
ex GiNaC::clifford::eval_ncmul (
    const exvector & v) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of clifford objects.

This removes superfluous ONEs, permutes gamma5/L/R's to the front and removes squares of gamma objects.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::class\\_info< OPT >::first](#), [clifford\(\)](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [get\\_commutator\\_sign\(\)](#), [GiNaC::hold\\_ncmul\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::reeval\\_ncmul\(\)](#), [representation\\_label](#), [same\\_metric\(\)](#), and [GiNaC::varidx::toggle\\_variance\(\)](#).

### 8.13.3.5 match\_same\_type()

```
bool GiNaC::clifford::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [clifford\(\)](#), [commutator\\_sign](#), [get\\_commutator\\_sign\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [representation\\_label](#), and [same\\_metric\(\)](#).

### 8.13.3.6 thiscontainer() [1/2]

```
ex GiNaC::clifford::thiscontainer (
    const exvector & v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [clifford\(\)](#), [commutator\\_sign](#), [metric](#), and [representation\\_label](#).

### 8.13.3.7 thiscontainer() [2/2]

```
ex GiNaC::clifford::thiscontainer (
    exvector && v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [clifford\(\)](#), [commutator\\_sign](#), [metric](#), and [representation\\_label](#).

### 8.13.3.8 return\_type()

```
unsigned GiNaC::clifford::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).



**8.13.3.9 return\_type\_tinfo()**

```
return_type_t GiNaC::clifford::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::make\\_return\\_type\\_t\(\)](#), and [representation\\_label](#).

**8.13.3.10 get\_representation\_label()**

```
unsigned char GiNaC::clifford::get_representation_label () const [inline]
```

References [representation\\_label](#).

Referenced by [GiNaC::cliffordunit::contract\\_with\(\)](#).

**8.13.3.11 get\_metric() [1/2]**

```
ex GiNaC::clifford::get_metric () const [inline]
```

References [metric](#).

Referenced by [GiNaC::cliffordunit::contract\\_with\(\)](#), and [same\\_metric\(\)](#).

**8.13.3.12 get\_metric() [2/2]**

```
ex GiNaC::clifford::get_metric (
    const ex & i,
    const ex & j,
    bool symmetrised = false) const [virtual]
```

References [GiNaC::\\_ex1\\_2](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::indexed::get\\_symmetry\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::is\\_a\(\)](#), [metric](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::indexed::simplify\\_indexed](#), and [GiNaC::symmetric2\(\)](#).

**8.13.3.13 same\_metric()**

```
bool GiNaC::clifford::same_metric (
    const ex & other) const
```

References [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [get\\_metric\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [op\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::indexed::simplify\\_indexed](#).

Referenced by [GiNaC::cliffordunit::contract\\_with\(\)](#), [eval\\_ncmul\(\)](#), and [match\\_same\\_type\(\)](#).

**8.13.3.14 get\_commutator\_sign()**

```
int GiNaC::clifford::get_commutator_sign () const [inline]
```

References [commutator\\_sign](#).

Referenced by [eval\\_ncmul\(\)](#), and [match\\_same\\_type\(\)](#).

**8.13.3.15 nops()**

```
size_t GiNaC::clifford::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

Referenced by [let\\_op\(\)](#), and [op\(\)](#).

**8.13.3.16 op()**

```
ex GiNaC::clifford::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

Referenced by [same\\_metric\(\)](#).

**8.13.3.17 let\_op()**

```
ex & GiNaC::clifford::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

**8.13.3.18 subs()**

```
ex GiNaC::clifford::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clifford\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [metric](#), and [GiNaC::ex::subs\(\)](#).

### 8.13.3.19 do\_print\_dflt()

```
void GiNaC::clifford::do_print_dflt (
    const print\_dflt & c,
    unsigned level) const [protected]
```

References [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), [GiNaC::indexed::printindices\(\)](#), [representation\\_label](#), [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.13.3.20 do\_print\_latex()

```
void GiNaC::clifford::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), [representation\\_label](#), [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.13.3.21 do\_print\_tree()

```
void GiNaC::clifford::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [metric](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::print\\_context::s](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::indexed::symtree](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.13.4 Member Data Documentation

### 8.13.4.1 representation\_label

```
unsigned char GiNaC::clifford::representation_label [protected]
```

Representation label to distinguish independent spin lines.

Referenced by [archive\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [eval\\_ncmul\(\)](#), [get\\_representation\\_label\(\)](#), [let\\_op\(\)](#), [match\\_same\\_type\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), [return\\_type\\_tinfo\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

### 8.13.4.2 metric

```
ex GiNaC::clifford::metric [protected]
```

Metric of the space, all constructors make it an indexed object.

Referenced by [archive\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_metric\(\)](#), [get\\_metric\(\)](#), [read\\_archive\(\)](#), [subs\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 8.13.4.3 commutator\_sign

```
int GiNaC::clifford::commutator_sign [protected]
```

It is the sign in the definition  $e_i e_j \pm e_j e_i = B(i, j) + B(j, i)$ .

Referenced by [archive\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [clifford\(\)](#), [get\\_commutator\\_sign\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

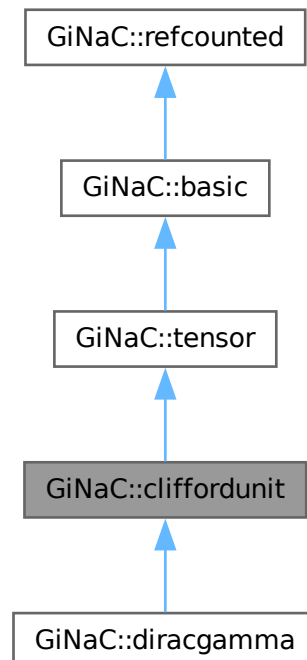
- [clifford.h](#)
- [clifford.cpp](#)

### 8.14 GiNaC::cliffordunit Class Reference

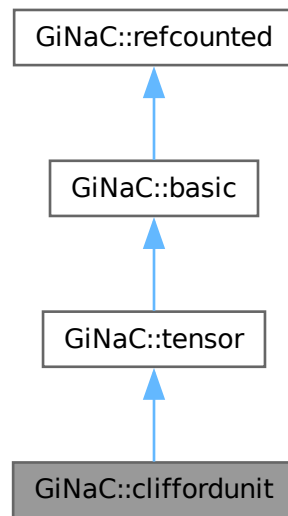
This class represents the Clifford algebra generators (units).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::cliffordunit:



Collaboration diagram for GiNaC::cliffordunit:



### Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of a Clifford unit with something else.*

### Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 8.14.1 Detailed Description

This class represents the Clifford algebra generators (units).



## 8.14.2 Member Function Documentation

### 8.14.2.1 `contract_with()`

```
bool GiNaC::cliffordunit::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of a Clifford unit with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::diracgamma](#).

References [GiNaC::\\_ex1](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::clifford::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::clifford::same\\_metric\(\)](#).

### 8.14.2.2 `do_print()`

```
void GiNaC::cliffordunit::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.14.2.3 `do_print_latex()`

```
void GiNaC::cliffordunit::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

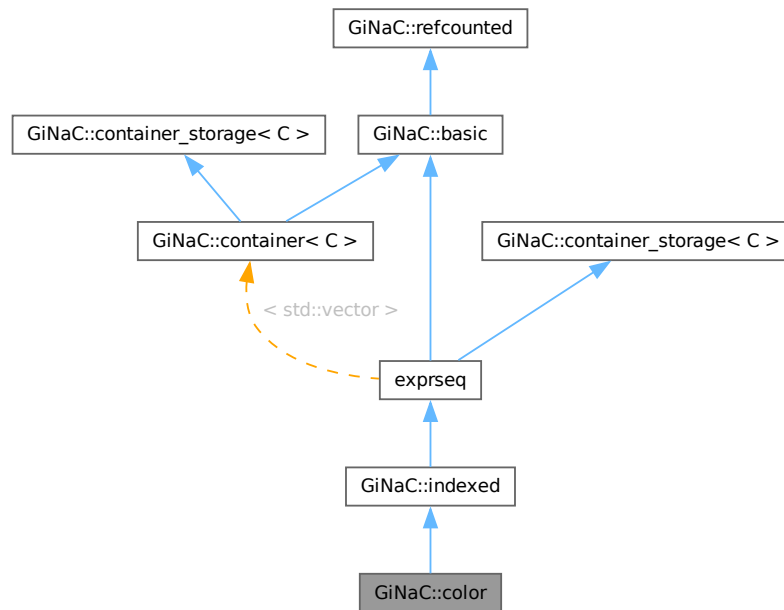
- [clifford.h](#)
- [clifford.cpp](#)

## 8.15 GiNaC::color Class Reference

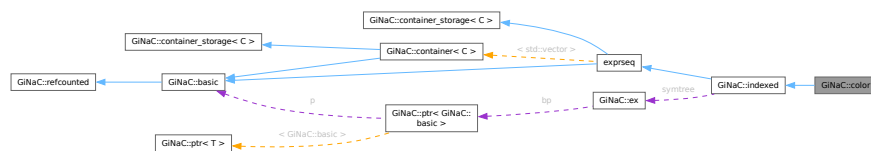
This class holds a generator `T_a` or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

```
#include <color.h>
```

Inheritance diagram for GiNaC::color:



Collaboration diagram for GiNaC::color:



## Public Member Functions

- **color** (const **ex** &b, unsigned char rl=0)  
*Construct object without any color index.*
- **color** (const **ex** &b, const **ex** &i1, unsigned char rl=0)  
*Construct object with one color index.*
- **color** (unsigned char rl, const **exvector** &v)
- **color** (unsigned char rl, **exvector** &&v)
- void **archive** (**archive\_node** &n) const override  
*Save (serialize) the object into archive node.*
- void **read\_archive** (const **archive\_node** &n, **lst** &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char **get\_representation\_label** () const

## Public Member Functions inherited from GiNaC::indexed

- `indexed` (const `ex` &b)  
*Construct indexed object with no index.*
- `indexed` (const `ex` &b, const `ex` &i1)  
*Construct indexed object with one index.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
*Construct indexed object with three indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
*Construct indexed object with four indices.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
*Construct indexed object with three indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
*Construct indexed object with four indices and a specified symmetry.*
- `indexed` (const `ex` &b, const `exvector` &iv)  
*Construct indexed object with a specified vector of indices.*
- `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)  
*Construct indexed object with a specified vector of indices and symmetry.*
- `indexed` (const `symmetry` &symm, const `exprseq` &es)
- `indexed` (const `symmetry` &symm, const `exvector` &v)
- `indexed` (const `symmetry` &symm, `exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- bool `all_index_values_are` (unsigned inf) const  
*Check whether all index values have a certain property.*
- `exvector get_indices` () const  
*Return a vector containing the object's indices.*
- `exvector get_dummy_indices` () const  
*Return a vector containing the dummy indices of the object, if any.*
- `exvector get_dummy_indices` (const `indexed` &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- bool `has_dummy_index_for` (const `ex` &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- `ex get_symmetry` () const  
*Return symmetry properties.*

## Public Member Functions inherited from [GiNaC::container< std::vector >](#)

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [container & prepend](#) (const [ex](#) &b)  
*Add element at front.*
- [container & append](#) (const [ex](#) &b)  
*Add element at back.*
- [container & remove\\_first](#) ()  
*Remove first element.*
- [container & remove\\_last](#) ()  
*Remove last element.*
- [container & remove\\_all](#) ()  
*Remove all elements.*
- [container & sort](#) ()  
*Sort elements.*
- [container & unique](#) ()  
*Remove adjacent duplicate elements.*
- [const\\_iterator begin](#) () const
- [const\\_iterator end](#) () const
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic & operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const

- *Output to stream.*  
virtual void [dbgprint](#) () const
- *Little wrapper around print to be called within a debugger.*  
virtual void [dbgprinttree](#) () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*

- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override  
*Perform automatic simplification on noncommutative product of color objects.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &v) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&v) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override

## Protected Member Functions inherited from `GiNaC::indexed`

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&v) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

### Protected Member Functions inherited from [GiNaC::container< std::vector >](#)

- [ex conjugate](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned options=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
[container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

### Private Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent color matrices coming from separated fermion lines.*

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< std::vector >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< std::vector >](#)

- typedef [container\\_storage< std::vector >::STLT](#) STLT

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > STLT

### Static Protected Member Functions inherited from [GiNaC::container< std::vector >](#)

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

### Protected Attributes inherited from [GiNaC::indexed](#)

- [ex](#) *symtree*  
*Index symmetry (tree of symmetry objects).*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- STLT *seq*



### 8.15.1 Detailed Description

This class holds a generator `T_a` or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Lie algebras (objects with different labels commute). These objects implement an abstract representation of the group, not a specific matrix representation. The indices used for color objects should not have a variance.

### 8.15.2 Constructor & Destructor Documentation

#### 8.15.2.1 `color()` [1/4]

```
GiNaC::color::color (
    const ex & b,
    unsigned char rl = 0)
```

Construct object without any color index.

This constructor is for internal use only. Use the `color_ONE()` function instead.

See also

[color\\_ONE](#)

References [color\(\)](#), and [representation\\_label](#).

Referenced by [color\(\)](#), [eval\\_ncmul\(\)](#), [match\\_same\\_type\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 8.15.2.2 `color()` [2/4]

```
GiNaC::color::color (
    const ex & b,
    const ex & il,
    unsigned char rl = 0)
```

Construct object with one color index.

This constructor is for internal use only. Use the `color_T()` function instead.

See also

[color\\_T](#)

References [representation\\_label](#).

#### 8.15.2.3 `color()` [3/4]

```
GiNaC::color::color (
    unsigned char rl,
    const exvector & v)
```

References [GiNaC::not\\_symmetric\(\)](#), and [representation\\_label](#).

#### 8.15.2.4 `color()` [4/4]

```
GiNaC::color::color (
    unsigned char rl,
    exvector && v)
```

References [GiNaC::not\\_symmetric\(\)](#), and [representation\\_label](#).

### 8.15.3 Member Function Documentation

#### 8.15.3.1 `archive()`

```
void GiNaC::color::archive (
    archive\_node & n) const \[override\], \[virtual\]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_unsigned\(\)](#), and [representation\\_label](#).

#### 8.15.3.2 `read_archive()`

```
void GiNaC::color::read_archive (
    const archive\_node & n,
    lst & syms) \[override\], \[virtual\]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::find\\_unsigned\(\)](#), and [representation\\_label](#).

### 8.15.3.3 eval\_ncmul()

```
ex GiNaC::color::eval_ncmul (
    const exvector & v) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of color objects.

This removes superfluous ONEs.

Reimplemented from [GiNaC::basic](#).

References [color\(\)](#), [eval\\_ncmul\(\)](#), [GiNaC::hold\\_ncmul\(\)](#), [GiNaC::is\\_a\(\)](#), and [representation\\_label](#).

Referenced by [eval\\_ncmul\(\)](#).

### 8.15.3.4 match\_same\_type()

```
bool GiNaC::color::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [color\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [representation\\_label](#).

### 8.15.3.5 thiscontainer() [1/2]

```
ex GiNaC::color::thiscontainer (
    const exvector & v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [color\(\)](#), and [representation\\_label](#).

### 8.15.3.6 thiscontainer() [2/2]

```
ex GiNaC::color::thiscontainer (
    exvector && v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [color\(\)](#), and [representation\\_label](#).

### 8.15.3.7 return\_type()

```
unsigned GiNaC::color::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

### 8.15.3.8 return\_type\_tinfo()

```
return_type_t GiNaC::color::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::make\\_return\\_type\\_t\(\)](#), and [representation\\_label](#).

### 8.15.3.9 get\_representation\_label()

```
unsigned char GiNaC::color::get_representation_label () const [inline]
```

References [representation\\_label](#).

## 8.15.4 Member Data Documentation

### 8.15.4.1 representation\_label

```
unsigned char GiNaC::color::representation_label [private]
```

Representation label to distinguish independent color matrices coming from separated fermion lines.

Referenced by [archive\(\)](#), [color\(\)](#), [color\(\)](#), [color\(\)](#), [color\(\)](#), [eval\\_ncmul\(\)](#), [get\\_representation\\_label\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [return\\_type\\_tinfo\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 8.16 GiNaC::compare\_all\_equal< T > Class Template Reference

Comparison policy: all structures of one type are equal.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_all\\_equal\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 8.16.1 Detailed Description

```
template<class T>
class GiNaC::compare_all_equal< T >
```

Comparison policy: all structures of one type are equal.

### 8.16.2 Constructor & Destructor Documentation

#### 8.16.2.1 ~compare\_all\_equal()

```
template<class T>
GiNaC::compare_all_equal< T >::~~compare_all_equal () [inline], [protected]
```

### 8.16.3 Member Function Documentation

#### 8.16.3.1 struct\_is\_equal()

```
template<class T>
bool GiNaC::compare_all_equal< T >::struct_is_equal (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

#### 8.16.3.2 struct\_compare()

```
template<class T>
int GiNaC::compare_all_equal< T >::struct_compare (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 8.17 GiNaC::compare\_bitwise< T > Class Template Reference

Comparison policy: use bit-wise comparison to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_bitwise\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 8.17.1 Detailed Description

```
template<class T>
class GiNaC::compare_bitwise< T >
```

Comparison policy: use bit-wise comparison to compare structures.

### 8.17.2 Constructor & Destructor Documentation

#### 8.17.2.1 ~compare\_bitwise()

```
template<class T>
GiNaC::compare_bitwise< T >::~~compare_bitwise () [inline], [protected]
```

### 8.17.3 Member Function Documentation

#### 8.17.3.1 struct\_is\_equal()

```
template<class T>
bool GiNaC::compare_bitwise< T >::struct_is_equal (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

#### 8.17.3.2 struct\_compare()

```
template<class T>
int GiNaC::compare_bitwise< T >::struct_compare (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 8.18 GiNaC::compare\_std\_less< T > Class Template Reference

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_std\\_less\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 8.18.1 Detailed Description

```
template<class T>
class GiNaC::compare_std_less< T >
```

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 ~compare\_std\_less()

```
template<class T>
GiNaC::compare_std_less< T >::~~compare_std_less () [inline], [protected]
```

### 8.18.3 Member Function Documentation

#### 8.18.3.1 struct\_is\_equal()

```
template<class T>
bool GiNaC::compare_std_less< T >::struct_is_equal (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

#### 8.18.3.2 struct\_compare()

```
template<class T>
int GiNaC::compare_std_less< T >::struct_compare (
    const T * t1,
    const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

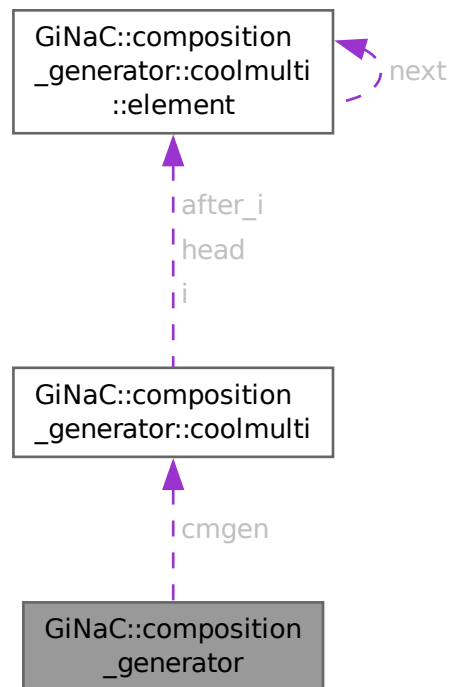
- [structure.h](#)

## 8.19 GiNaC::composition\_generator Class Reference

Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.

```
#include <utils.h>
```

Collaboration diagram for GiNaC::composition\_generator:



### Classes

- struct [coolmulti](#)

### Public Member Functions

- [composition\\_generator](#) (const std::vector< unsigned > &partition)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

### Private Attributes

- struct [GiNaC::composition\\_generator::coolmulti](#) cmgen
- bool [atend](#)
- bool [trivial](#)
- std::vector< unsigned > [composition](#)
- bool [current\\_updated](#)



### 8.19.1 Detailed Description

Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.

### 8.19.2 Constructor & Destructor Documentation

#### 8.19.2.1 composition\_generator()

```
GiNaC::composition_generator::composition_generator (
    const std::vector< unsigned > & partition) [inline], [explicit]
```

References [atend](#), [cmgen](#), [composition](#), [current\\_updated](#), and [trivial](#).

### 8.19.3 Member Function Documentation

#### 8.19.3.1 get()

```
const std::vector< unsigned > & GiNaC::composition_generator::get () const [inline]
```

References [cmgen](#), [composition](#), [current\\_updated](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::value](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

#### 8.19.3.2 next()

```
bool GiNaC::composition_generator::next () [inline]
```

References [atend](#), [cmgen](#), [current\\_updated](#), and [trivial](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

### 8.19.4 Member Data Documentation

#### 8.19.4.1 cmgen

```
struct GiNaC::composition_generator::coolmulti GiNaC::composition_generator::cmgen [private]
```

Referenced by [composition\\_generator\(\)](#), [get\(\)](#), and [next\(\)](#).

#### 8.19.4.2 atend

```
bool GiNaC::composition_generator::atend [private]
```

Referenced by [composition\\_generator\(\)](#), and [next\(\)](#).

#### 8.19.4.3 trivial

```
bool GiNaC::composition_generator::trivial [private]
```

Referenced by [composition\\_generator\(\)](#), and [next\(\)](#).

#### 8.19.4.4 composition

```
std::vector<unsigned> GiNaC::composition_generator::composition [mutable], [private]
```

Referenced by [composition\\_generator\(\)](#), and [get\(\)](#).

#### 8.19.4.5 current\_updated

```
bool GiNaC::composition_generator::current_updated [mutable], [private]
```

Referenced by [composition\\_generator\(\)](#), [get\(\)](#), and [next\(\)](#).

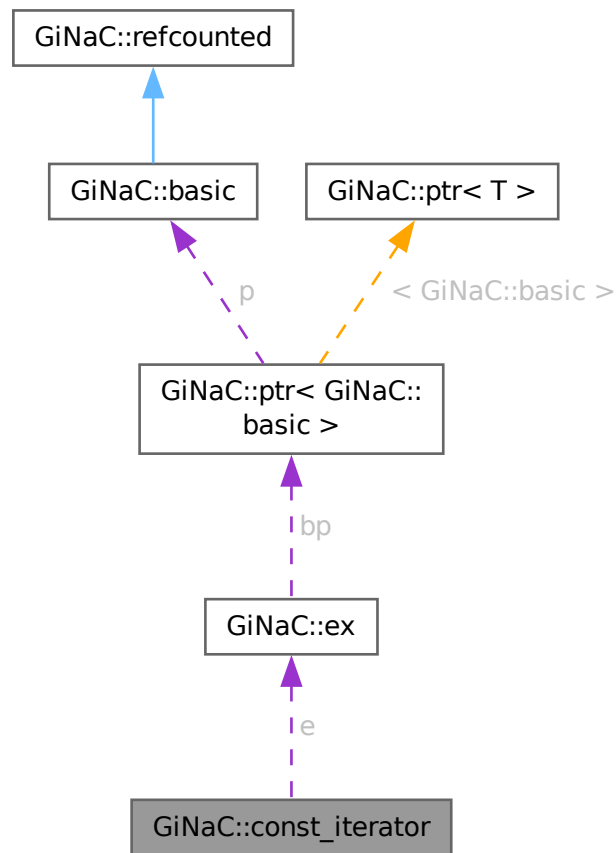
The documentation for this class was generated from the following file:

- [utils.h](#)

## 8.20 GiNaC::const\_iterator Class Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::const\_iterator:



## Public Types

- using `iterator_category` = `std::random_access_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

## Public Member Functions

- `const_iterator` () noexcept
- `ex operator*` () const
- `std::unique_ptr< ex > operator->` () const
- `ex operator[]` (`difference_type` n) const
- `const_iterator & operator++` () noexcept
- `const_iterator operator++` (int) noexcept
- `const_iterator & operator+=` (`difference_type` n) noexcept
- `const_iterator operator+` (`difference_type` n) const noexcept

- `const_iterator` & `operator--` () noexcept
- `const_iterator` `operator--` (int) noexcept
- `const_iterator` & `operator-=` (`difference_type` n) noexcept
- `const_iterator` `operator-` (`difference_type` n) const noexcept
- bool `operator==` (const `const_iterator` &other) const noexcept
- bool `operator!=` (const `const_iterator` &other) const noexcept
- bool `operator<` (const `const_iterator` &other) const noexcept
- bool `operator>` (const `const_iterator` &other) const noexcept
- bool `operator<=` (const `const_iterator` &other) const noexcept
- bool `operator>=` (const `const_iterator` &other) const noexcept

### Protected Attributes

- `ex` `e`
- `size_t` `i`

### Private Member Functions

- `const_iterator` (const `ex` &`e_`, `size_t` `i_`) noexcept

### Friends

- class `ex`
- class `const_preorder_iterator`
- class `const_postorder_iterator`
- `const_iterator` `operator+` (`difference_type` n, const `const_iterator` &it) noexcept
- `difference_type` `operator-` (const `const_iterator` &`lhs`, const `const_iterator` &`rhs`) noexcept

## 8.20.1 Member Typedef Documentation

### 8.20.1.1 iterator\_category

```
using GiNaC::const_iterator::iterator_category = std::random_access_iterator_tag
```

### 8.20.1.2 value\_type

```
using GiNaC::const_iterator::value_type = ex
```

### 8.20.1.3 difference\_type

```
using GiNaC::const_iterator::difference_type = ptrdiff_t
```

### 8.20.1.4 pointer

```
using GiNaC::const_iterator::pointer = const ex *
```

### 8.20.1.5 reference

using [GiNaC::const\\_iterator::reference](#) = const [ex](#) &

## 8.20.2 Constructor & Destructor Documentation

### 8.20.2.1 const\_iterator() [1/2]

[GiNaC::const\\_iterator::const\\_iterator](#) () [inline], [noexcept]

Referenced by [operator!=\(\)](#), [operator+\(\)](#), [operator+](#), [operator++\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-](#), [operator-\(\)](#), [operator--\(\)](#), [operator--\(\)](#), [operator-=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), and [operator>=\(\)](#).

### 8.20.2.2 const\_iterator() [2/2]

```
GiNaC::const\_iterator::const\_iterator (
    const ex & e_,
    size_t i_) [inline], [private], [noexcept]
```

References [ex](#).

## 8.20.3 Member Function Documentation

### 8.20.3.1 operator\*()

[ex](#) [GiNaC::const\\_iterator::operator\\*](#) () const [inline]

References [e](#), [ex](#), and [i](#).

### 8.20.3.2 operator->()

[std::unique\\_ptr](#)< [ex](#) > [GiNaC::const\\_iterator::operator->](#) () const [inline]

References [ex](#).

### 8.20.3.3 operator[]()

```
ex GiNaC::const\_iterator::operator\[\] (
    difference\_type n) const [inline]
```

References [e](#), [ex](#), and [i](#).

### 8.20.3.4 operator++() [1/2]

[const\\_iterator](#) & [GiNaC::const\\_iterator::operator++](#) () [inline], [noexcept]

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.5 operator++() [2/2]

```
const_iterator GiNaC::const_iterator::operator++ (
    int ) [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.6 operator+=()

```
const_iterator & GiNaC::const_iterator::operator+= (
    difference_type n) [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.7 operator+()

```
const_iterator GiNaC::const_iterator::operator+ (
    difference_type n) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), and [i](#).

#### 8.20.3.8 operator--() [1/2]

```
const_iterator & GiNaC::const_iterator::operator-- () [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.9 operator--() [2/2]

```
const_iterator GiNaC::const_iterator::operator-- (
    int ) [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.10 operator-=()

```
const_iterator & GiNaC::const_iterator::operator-= (
    difference_type n) [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

#### 8.20.3.11 operator-()

```
const_iterator GiNaC::const_iterator::operator- (
    difference_type n) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), and [i](#).

### 8.20.3.12 operator==( )

```
bool GiNaC::const_iterator::operator== (
    const const\_iterator & other) const [inline], [noexcept]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [const\\_iterator\(\)](#), [e](#), and [i](#).

### 8.20.3.13 operator!=( )

```
bool GiNaC::const_iterator::operator!= (
    const const\_iterator & other) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#).

### 8.20.3.14 operator<( )

```
bool GiNaC::const_iterator::operator< (
    const const\_iterator & other) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), and [i](#).

### 8.20.3.15 operator>( )

```
bool GiNaC::const_iterator::operator> (
    const const\_iterator & other) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#).

### 8.20.3.16 operator<=( )

```
bool GiNaC::const_iterator::operator<= (
    const const\_iterator & other) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#).

### 8.20.3.17 operator>=( )

```
bool GiNaC::const_iterator::operator>= (
    const const\_iterator & other) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#).

## 8.20.4 Friends And Related Symbol Documentation

### 8.20.4.1 ex

```
friend class ex [friend]
```

References [ex](#).

Referenced by [const\\_iterator\(\)](#), [ex](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator\[\]\(\)](#).

#### 8.20.4.2 `const_preorder_iterator`

```
friend class const_preorder_iterator [friend]
```

References [const\\_preorder\\_iterator](#).

Referenced by [const\\_preorder\\_iterator](#).

#### 8.20.4.3 `const_postorder_iterator`

```
friend class const_postorder_iterator [friend]
```

References [const\\_postorder\\_iterator](#).

Referenced by [const\\_postorder\\_iterator](#).

#### 8.20.4.4 `operator+`

```
const_iterator operator+ (
    difference_type n,
    const const_iterator & it) [friend]
```

References [const\\_iterator\(\)](#).

#### 8.20.4.5 `operator-`

```
difference_type operator- (
    const const_iterator & lhs,
    const const_iterator & rhs) [friend]
```

References [const\\_iterator\(\)](#), [GiNaC::lhs\(\)](#), and [GiNaC::rhs\(\)](#).

### 8.20.5 Member Data Documentation

#### 8.20.5.1 `e`

```
ex GiNaC::const_iterator::e [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator-\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

#### 8.20.5.2 `i`

```
size_t GiNaC::const_iterator::i [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator-=\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)



## 8.21 GiNaC::const\_postorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using `iterator_category` = `std::forward_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

### Public Member Functions

- `const_postorder_iterator` () noexcept
- `const_postorder_iterator` (const `ex` &`e`, `size_t` `n`)
- `reference operator*` () const
- `pointer operator->` () const
- `const_postorder_iterator & operator++` ()
- `const_postorder_iterator operator++` (int)
- bool `operator==` (const `const_postorder_iterator` &`other`) const noexcept
- bool `operator!=` (const `const_postorder_iterator` &`other`) const noexcept

### Private Member Functions

- void `descend` ()
- void `increment` ()

### Private Attributes

- `std::stack< internal::_iter_rep, std::vector< internal::_iter_rep > >` `s`

## 8.21.1 Member Typedef Documentation

### 8.21.1.1 iterator\_category

```
using GiNaC::const_postorder_iterator::iterator_category = std::forward_iterator_tag
```

### 8.21.1.2 value\_type

```
using GiNaC::const_postorder_iterator::value_type = ex
```

### 8.21.1.3 difference\_type

```
using GiNaC::const_postorder_iterator::difference_type = ptrdiff_t
```

#### 8.21.1.4 pointer

```
using GiNaC::const_postorder_iterator::pointer = const ex *
```

#### 8.21.1.5 reference

```
using GiNaC::const_postorder_iterator::reference = const ex &
```

### 8.21.2 Constructor & Destructor Documentation

#### 8.21.2.1 const\_postorder\_iterator() [1/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator () [inline], [noexcept]
```

Referenced by [operator!=\(\)](#), [operator++\(\)](#), [operator++\(\)](#), and [operator==\(\)](#).

#### 8.21.2.2 const\_postorder\_iterator() [2/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator (
    const ex & e,
    size_t n) [inline]
```

References [descend\(\)](#), and [s](#).

### 8.21.3 Member Function Documentation

#### 8.21.3.1 operator\*()

```
reference GiNaC::const_postorder_iterator::operator* () const [inline]
```

References [s](#).

#### 8.21.3.2 operator->()

```
pointer GiNaC::const_postorder_iterator::operator-> () const [inline]
```

References [s](#).

#### 8.21.3.3 operator++() [1/2]

```
const\_postorder\_iterator & GiNaC::const_postorder_iterator::operator++ () [inline]
```

References [const\\_postorder\\_iterator\(\)](#), and [increment\(\)](#).

#### 8.21.3.4 operator++() [2/2]

```
const_postorder_iterator GiNaC::const_postorder_iterator::operator++ (
    int ) [inline]
```

References [const\\_postorder\\_iterator\(\)](#), and [increment\(\)](#).

#### 8.21.3.5 operator==( )

```
bool GiNaC::const_postorder_iterator::operator== (
    const const_postorder_iterator & other) const [inline], [noexcept]
```

References [const\\_postorder\\_iterator\(\)](#), and [s](#).

#### 8.21.3.6 operator!=( )

```
bool GiNaC::const_postorder_iterator::operator!= (
    const const_postorder_iterator & other) const [inline], [noexcept]
```

References [const\\_postorder\\_iterator\(\)](#).

#### 8.21.3.7 descend()

```
void GiNaC::const_postorder_iterator::descend () [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [const\\_postorder\\_iterator\(\)](#), and [increment\(\)](#).

#### 8.21.3.8 increment()

```
void GiNaC::const_postorder_iterator::increment () [inline], [private]
```

References [descend\(\)](#), and [s](#).

Referenced by [operator++\(\)](#), and [operator++\(\)](#).

### 8.21.4 Member Data Documentation

#### 8.21.4.1 s

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_postorder_↵
iterator::s [private]
```

Referenced by [const\\_postorder\\_iterator\(\)](#), [descend\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\( \)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)

## 8.22 GiNaC::const\_preorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using `iterator_category` = `std::forward_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

### Public Member Functions

- `const_preorder_iterator` () noexcept
- `const_preorder_iterator` (const `ex` &`e`, `size_t` `n`)
- `reference operator*` () const
- `pointer operator->` () const
- `const_preorder_iterator & operator++` ()
- `const_preorder_iterator operator++` (int)
- `bool operator==` (const `const_preorder_iterator` &`other`) const noexcept
- `bool operator!=` (const `const_preorder_iterator` &`other`) const noexcept

### Private Member Functions

- void `increment` ()

### Private Attributes

- `std::stack< internal::_iter_rep, std::vector< internal::_iter_rep > >` `s`

## 8.22.1 Member Typedef Documentation

### 8.22.1.1 iterator\_category

```
using GiNaC::const_preorder_iterator::iterator_category = std::forward_iterator_tag
```

### 8.22.1.2 value\_type

```
using GiNaC::const_preorder_iterator::value_type = ex
```

### 8.22.1.3 difference\_type

```
using GiNaC::const_preorder_iterator::difference_type = ptrdiff_t
```

#### 8.22.1.4 pointer

using [GiNaC::const\\_preorder\\_iterator::pointer](#) = const [ex](#) \*

#### 8.22.1.5 reference

using [GiNaC::const\\_preorder\\_iterator::reference](#) = const [ex](#) &

### 8.22.2 Constructor & Destructor Documentation

#### 8.22.2.1 const\_preorder\_iterator() [1/2]

[GiNaC::const\\_preorder\\_iterator::const\\_preorder\\_iterator](#) () [inline], [noexcept]

Referenced by [operator!\(\)](#), [operator++\(\)](#), [operator++\(\)](#), and [operator==\(\)](#).

#### 8.22.2.2 const\_preorder\_iterator() [2/2]

```
GiNaC::const\_preorder\_iterator::const\_preorder\_iterator (  
    const ex & e,  
    size_t n) [inline]
```

References [s](#).

### 8.22.3 Member Function Documentation

#### 8.22.3.1 operator\*()

[reference](#) [GiNaC::const\\_preorder\\_iterator::operator\\*](#) () const [inline]

References [s](#).

#### 8.22.3.2 operator->()

[pointer](#) [GiNaC::const\\_preorder\\_iterator::operator->](#) () const [inline]

References [s](#).

#### 8.22.3.3 operator++() [1/2]

[const\\_preorder\\_iterator](#) & [GiNaC::const\\_preorder\\_iterator::operator++](#) () [inline]

References [const\\_preorder\\_iterator\(\)](#), and [increment\(\)](#).

### 8.22.3.4 `operator++()` [2/2]

```
const_preorder_iterator GiNaC::const_preorder_iterator::operator++ (
    int ) [inline]
```

References [const\\_preorder\\_iterator\(\)](#), and [increment\(\)](#).

### 8.22.3.5 `operator==()`

```
bool GiNaC::const_preorder_iterator::operator== (
    const const_preorder_iterator & other) const [inline], [noexcept]
```

References [const\\_preorder\\_iterator\(\)](#), and [s](#).

### 8.22.3.6 `operator!=(())`

```
bool GiNaC::const_preorder_iterator::operator!= (
    const const_preorder_iterator & other) const [inline], [noexcept]
```

References [const\\_preorder\\_iterator\(\)](#).

### 8.22.3.7 `increment()`

```
void GiNaC::const_preorder_iterator::increment () [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::internal::\\_iter\\_rep::i\\_end](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [operator++\(\)](#), and [operator++\(\)](#).

## 8.22.4 Member Data Documentation

### 8.22.4.1 `s`

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_preorder_iterator::s [private]
```

Referenced by [const\\_preorder\\_iterator\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following file:

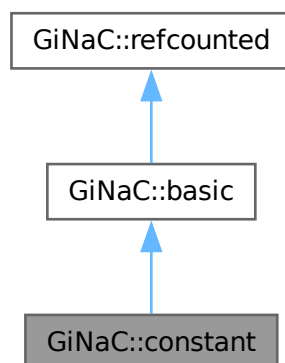
- [ex.h](#)

## 8.23 GiNaC::constant Class Reference

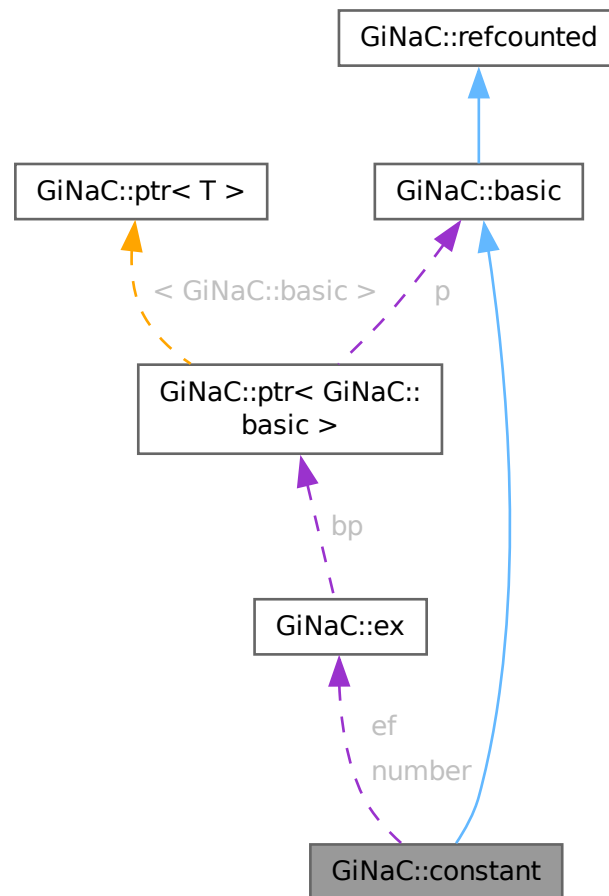
This class holds constants, symbols with specific numerical value.

```
#include <constant.h>
```

Inheritance diagram for GiNaC::constant:



Collaboration diagram for `GiNaC::constant`:



## Public Member Functions

- `constant` (const std::string &initname, `evalffunctype` efun=nullptr, const std::string &texname=std::string(), unsigned `domain=domain::complex`)
- `constant` (const std::string &initname, const `numeric` &initnumber, const std::string &texname=std::string(), unsigned `domain=domain::complex`)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*



- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Load (deserialize) the object from an archive node.*
- std::string `get_name` () const
- std::string `get_TeX_name` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position `i`.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position `i`.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual int `degree` (const `ex` &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*

  - unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*

  - const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a constant always returns 0.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex\\_eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- std::string [name](#)  
*printrname of this constant*
- std::string [TeX\\_name](#)  
*LaTeX name.*
- [evalffunctype](#) ef
- [ex\\_number](#)  
*numerical value this constant [evalf\(\)](#)s to*
- unsigned [serial](#)  
*unique serial number for comparison*
- unsigned [domain](#)  
*numerical value this constant [evalf\(\)](#)s to*

### Static Private Attributes

- static unsigned [next\\_serial](#) = 0

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 8.23.1 Detailed Description

This class holds constants, symbols with specific numerical value.

Each object of this class must either provide their own function to evaluate it to class numeric or provide the constant as a numeric (if it's an exact number).

## 8.23.2 Constructor & Destructor Documentation

### 8.23.2.1 [constant\(\)](#) [1/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    evalffunctype efun = nullptr,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex)
```

References [constant\(\)](#), [domain](#), [ef](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

Referenced by [constant\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

### 8.23.2.2 [constant\(\)](#) [2/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    const numeric & initnumber,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex)
```

References [domain](#), [ef](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [next\\_serial](#), [number](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

### 8.23.3 Member Function Documentation

#### 8.23.3.1 info()

```
bool GiNaC::constant::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [domain](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::domain::real](#), and [GiNaC::info\\_flags::real](#).

#### 8.23.3.2 evalf()

```
ex GiNaC::constant::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [ef](#), [GiNaC::basic::ex](#), and [number](#).

#### 8.23.3.3 is\_polynomial()

```
bool GiNaC::constant::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

#### 8.23.3.4 conjugate()

```
ex GiNaC::constant::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [domain](#), [GiNaC::basic::ex](#), [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 8.23.3.5 `real_part()`

```
ex GiNaC::constant::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [domain](#), [GiNaC::basic::ex](#), [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 8.23.3.6 `imag_part()`

```
ex GiNaC::constant::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [domain](#), [GiNaC::basic::ex](#), [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 8.23.3.7 `archive()`

```
void GiNaC::constant::archive (
    archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_string\(\)](#), and [name](#).

### 8.23.3.8 `read_archive()`

```
void GiNaC::constant::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::Catalan](#), [GiNaC::Euler](#), [GiNaC::archive\\_node::find\\_string\(\)](#), and [GiNaC::Pi](#).

### 8.23.3.9 derivative()

```
ex GiNaC::constant::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a constant always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and [GiNaC::basic::ex](#).

### 8.23.3.10 is\_equal\_same\_type()

```
bool GiNaC::constant::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [constant\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [serial](#).

### 8.23.3.11 calchash()

```
unsigned GiNaC::constant::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class [basic](#) computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 8.23.3.12 get\_name()

```
std::string GiNaC::constant::get_name () const [inline]
```

References [name](#).

#### 8.23.3.13 `get_TeX_name()`

```
std::string GiNaC::constant::get_TeX_name () const [inline]
```

References [TeX\\_name](#).

#### 8.23.3.14 `do_print()`

```
void GiNaC::constant::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [name](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.23.3.15 `do_print_tree()`

```
void GiNaC::constant::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [name](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.23.3.16 `do_print_latex()`

```
void GiNaC::constant::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_context::s](#), and [TeX\\_name](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.23.3.17 `do_print_python_repr()`

```
void GiNaC::constant::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [name](#), [GiNaC::print\\_context::s](#), and [TeX\\_name](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).



## 8.23.4 Member Data Documentation

### 8.23.4.1 name

```
std::string GiNaC::constant::name [private]
```

printname of this constant

Referenced by [archive\(\)](#), [constant\(\)](#), [constant\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), and [get\\_name\(\)](#).

### 8.23.4.2 TeX\_name

```
std::string GiNaC::constant::TeX_name [private]
```

LaTeX name.

Referenced by [constant\(\)](#), [constant\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), and [get\\_TeX\\_name\(\)](#).

### 8.23.4.3 ef

```
evalffunctype GiNaC::constant::ef [private]
```

Referenced by [constant\(\)](#), [constant\(\)](#), and [evalf\(\)](#).

### 8.23.4.4 number

```
ex GiNaC::constant::number [private]
```

numerical value this constant [evalf\(\)](#)s to

Referenced by [constant\(\)](#), and [evalf\(\)](#).

### 8.23.4.5 serial

```
unsigned GiNaC::constant::serial [private]
```

unique serial number for comparison

Referenced by [calchash\(\)](#), [constant\(\)](#), [constant\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

### 8.23.4.6 next\_serial

```
unsigned GiNaC::constant::next_serial = 0 [static], [private]
```

Referenced by [constant\(\)](#), and [constant\(\)](#).

### 8.23.4.7 domain

```
unsigned GiNaC::constant::domain [private]
```

numerical value this constant [evalf\(\)](#)s to

Referenced by [conjugate\(\)](#), [constant\(\)](#), [constant\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), and [real\\_part\(\)](#).

The documentation for this class was generated from the following files:

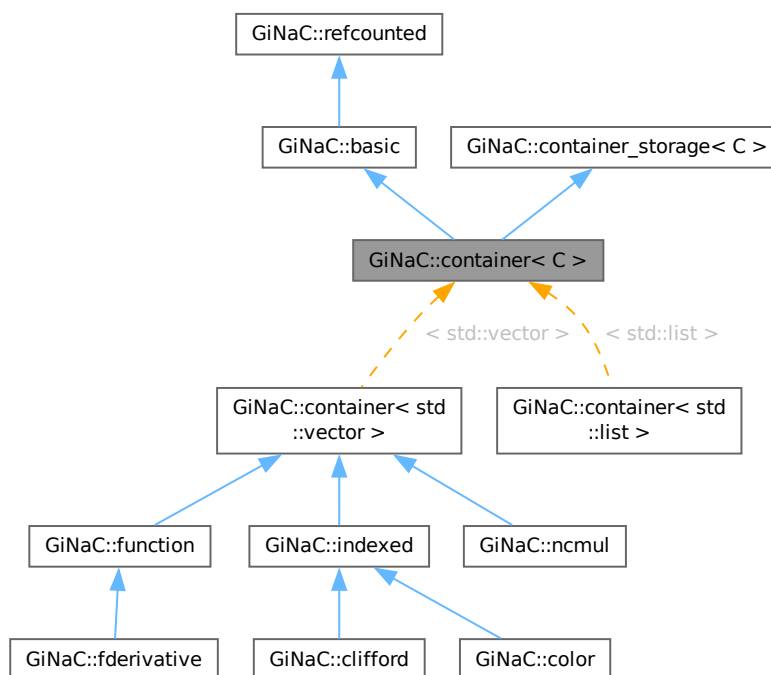
- [constant.h](#)
- [constant.cpp](#)

## 8.24 GiNaC::container< C > Class Template Reference

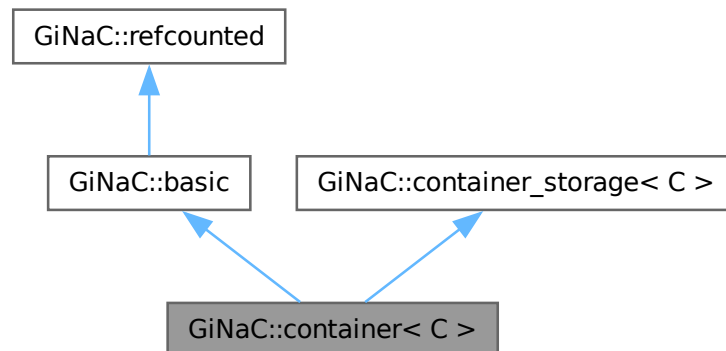
Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include <container.h>
```

Inheritance diagram for `GiNaC::container< C >`:



Collaboration diagram for GiNaC::container< C >:



## Public Types

- typedef `STLT::const_iterator` `const_iterator`
- typedef `STLT::const_reverse_iterator` `const_reverse_iterator`

## Public Member Functions

- `container` (`STLT const &s`)
- `container` (`STLT &&v`)
- `container` (`exvector::const_iterator b`, `exvector::const_iterator e`)
- `container` (`std::initializer_list< ex > il`)
- `bool info` (`unsigned inf`) `const` override  
*Information about the object.*
- `unsigned precedence` () `const` override  
*Return relative operator precedence (for parenthezing output).*
- `size_t nops` () `const` override  
*Number of operands/members.*
- `ex op` (`size_t i`) `const` override  
*Return operand/member at position i.*
- `ex & let_op` (`size_t i`) `override`  
*Return modifiable operand/member at position i.*
- `ex subs` (`const exmap &m`, `unsigned options=0`) `const` override  
*Substitute a set of objects by arbitrary expressions.*
- `void read_archive` (`const archive_node &n`, `lst &sym_lst`) `override`  
*Load (deserialize) the object from an archive node.*
- `void archive` (`archive_node &n`) `const` override  
*Archive the object.*
- `container & prepend` (`const ex &b`)  
*Add element at front.*
- `container & append` (`const ex &b`)  
*Add element at back.*

- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprntree () const`  
*Little wrapper around `prntree` to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`

- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex` `coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex` `expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex` `collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex` `series` (const `relational` &r, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex` `normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex` `to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex` `to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex` `smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric` `max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector` `get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex` `add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex` `scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex` `subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex` `diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Types

- typedef [container\\_storage](#)< C >::STLT [STLT](#)

## Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > [STLT](#)

## Protected Member Functions

- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this↔  
\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned options=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

**Static Protected Member Functions**

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) (STLT &, size\_t)

**Private Member Functions**

- void [sort\\_](#) (std::random\_access\_iterator\_tag)
  - void [sort\\_](#) (std::input\_iterator\_tag)
  - void [unique\\_](#) ()
  - void [unique\\_](#) ()
- Specialization of [container::unique\\_\(\)](#) for std::list.*

**Additional Inherited Members****Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
  of type [status\\_flags](#)
- unsigned [hashvalue](#)  
  hash value

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**8.24.1 Detailed Description**

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container< C >
```

Wrapper template for making [GiNaC](#) classes out of STL containers.

## 8.24.2 Member Typedef Documentation

### 8.24.2.1 STL

```
template<template< class T, class=std::allocator< T > > class C>
typedef container\_storage<C>::STLT GiNaC::container< C >::STLT [protected]
```

### 8.24.2.2 const\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STL::const_iterator GiNaC::container< C >::const_iterator
```

### 8.24.2.3 const\_reverse\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STL::const_reverse_iterator GiNaC::container< C >::const_reverse_iterator
```

## 8.24.3 Constructor & Destructor Documentation

### 8.24.3.1 container() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    STL const & s) [inline]
```

Referenced by [append\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [prepend\(\)](#), [remove\\_all\(\)](#), [remove\\_first\(\)](#), [remove\\_last\(\)](#), [sort\(\)](#), [subs\(\)](#), [GiNaC::container< std::vector >::thiscontainer\(\)](#), [GiNaC::container< std::vector >::thiscontainer\(\)](#), and [unique\(\)](#).

### 8.24.3.2 container() [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    STL && v) [inline], [explicit]
```

### 8.24.3.3 container() [3/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    exvector::const_iterator b,
    exvector::const_iterator e) [inline]
```

### 8.24.3.4 container() [4/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    std::initializer_list< ex > il) [inline]
```



## 8.24.4 Member Function Documentation

### 8.24.4.1 get\_default\_flags()

```
template<template< class T, class=std::allocator< T > > class C>
unsigned GiNaC::container< C >::get_default_flags () [inline], [static], [protected]
```

Referenced by [GiNaC::container< std::vector >::container\(\)](#), [GiNaC::container< std::vector >::container\(\)](#), [GiNaC::container< std::vector >::container\(\)](#), [GiNaC::container< std::vector >::container\(\)](#), and [GiNaC::container< std::vector >::re](#)

### 8.24.4.2 get\_open\_delim()

```
template<template< class T, class=std::allocator< T > > class C>
char GiNaC::container< C >::get_open_delim () [inline], [static], [protected]
```

Referenced by [do\\_print\(\)](#).

### 8.24.4.3 get\_close\_delim()

```
template<template< class T, class=std::allocator< T > > class C>
char GiNaC::container< C >::get_close_delim () [inline], [static], [protected]
```

Referenced by [do\\_print\(\)](#).

### 8.24.4.4 info()

```
template<template< class T, class=std::allocator< T > > class C>
template bool GiNaC::container< C >::info (
    unsigned inf) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

### 8.24.4.5 precedence()

```
template<template< class T, class=std::allocator< T > > class C>
unsigned GiNaC::container< C >::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_python\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

#### 8.24.4.6 nops()

```
template<template< class T, class=std::allocator< T > > class C>
size_t GiNaC::container< C >::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

Referenced by [GiNaC::ex::denom\(\)](#), [GiNaC::diag\\_matrix\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Gt::kernel::kernel\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::lst\\_to\\_matrix\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 8.24.4.7 op()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::nops\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::ex::denom\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::Gt::kernel::kernel\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

#### 8.24.4.8 let\_op()

```
template<template< class T, class=std::allocator< T > > class C>
ex & GiNaC::container< C >::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::nops\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::H\\_evalf\(\)](#).

#### 8.24.4.9 subs()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [container\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [subschildren\(\)](#), and [thiscontainer\(\)](#).

#### 8.24.4.10 read\_archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::read_archive (
    const archive_node & n,
    lst & syms) [inline], [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

#### 8.24.4.11 archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::archive (
    archive_node & n) const [inline], [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

#### 8.24.4.12 conjugate()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::conjugate () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), and [GiNaC::ncmul](#).

#### 8.24.4.13 real\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::real_part () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

#### 8.24.4.14 imag\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::imag_part () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

#### 8.24.4.15 is\_equal\_same\_type()

```
template<template< class T, class=std::allocator< T > > class C>
bool GiNaC::container< C >::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), and [GiNaC::function](#).

References [GiNaC::basic::basic\(\)](#), [container\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.24.4.16 thiscontainer() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
    const STLT & v) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

Referenced by [GiNaC::container< std::vector >::conjugate\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), and [subs\(\)](#).

**8.24.4.17 thiscontainer() [2/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
    STLT && v) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

**8.24.4.18 printseq()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::printseq (
    const print_context & c,
    char openbracket,
    char delim,
    char closebracket,
    unsigned this_precedence,
    unsigned upper_precedence = 0) const [protected], [virtual]
```

Print sequence of contained elements.

References [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_python\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

**8.24.4.19 sort\_() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
    std::random_access_iterator_tag ) [inline], [private]
```

Referenced by [sort\(\)](#).

**8.24.4.20 sort\_()** [2/2]

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
    std::input_iterator_tag ) [inline], [private]
```

**8.24.4.21 unique\_()** [1/2]

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::unique_ () [inline], [private]
```

Referenced by [unique\(\)](#).

**8.24.4.22 prepend()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::prepend (
    const ex & b)
```

Add element at front.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.24.4.23 append()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::append (
    const ex & b)
```

Add element at back.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::Gt::archive\(\)](#), [GiNaC::clifford\\_to\\_lst\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::G\\_path\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::ifactor\(\)](#), [GiNaC::Gt\\_detail::TransformExpressionWithCache< Type >::impl\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**8.24.4.24 remove\_first()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_first ()
```

Remove first element.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::H\\_deriv\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**8.24.4.25 remove\_last()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_last ()
```

Remove last element.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.24.4.26 remove\_all()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_all ()
```

Remove all elements.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::clifford\\_to\\_lst\(\)](#).

**8.24.4.27 sort()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::sort ()
```

Sort elements.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [sort\\_\(\)](#).

**8.24.4.28 unique()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::unique ()
```

Remove adjacent duplicate elements.

References [container\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [unique\\_\(\)](#).

**8.24.4.29 begin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::begin () const [inline]
```

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [GiNaC::ex::symmetrize\\_cyclic\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

**8.24.4.30 end()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::end () const [inline]
```

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), [GiNaC::ex::subs\(\)](#), [subschildren\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [GiNaC::ex::symmetrize\\_cyclic\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

**8.24.4.31 rbegin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rbegin () const [inline]
```

**8.24.4.32 rend()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rend () const [inline]
```

**8.24.4.33 do\_print()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [get\\_close\\_delim\(\)](#), [get\\_open\\_delim\(\)](#), [precedence\(\)](#), and [printseq\(\)](#).

**8.24.4.34 do\_print\_tree()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_tree (
    const print_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [end\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::nops\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.24.4.35 do\_print\_python()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python (
    const print_python & c,
    unsigned level) const [protected]
```

References [precedence\(\)](#), and [printseq\(\)](#).



## 8.24.4.36 do\_print\_python\_repr()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python_repr (
    const print_python_repr & c,
    unsigned level) const [protected]
```

References [precedence\(\)](#), [printseq\(\)](#), and [GiNaC::print\\_context::s](#).

## 8.24.4.37 subschildren()

```
template<template< class T, class=std::allocator< T > > class C>
container< C >::STLT GiNaC::container< C >::subschildren (
    const exmap & m,
    unsigned options = 0) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [end\(\)](#), [GiNaC::basic::ex](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [subs\(\)](#).

## 8.24.4.38 unique\_() [2/2]

```
void GiNaC::container< std::list >::unique_ () [inline], [private]
```

Specialization of [container::unique\\_\(\)](#) for `std::list`.

The documentation for this class was generated from the following files:

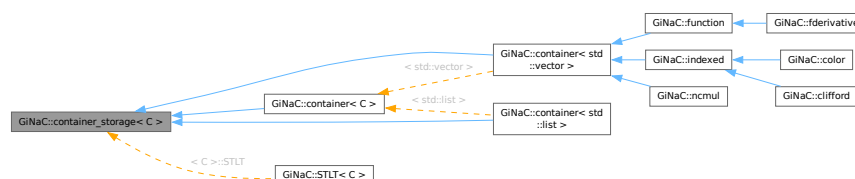
- [container.h](#)
- [lst.h](#)

## 8.25 GiNaC::container\_storage&lt; C &gt; Class Template Reference

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

```
#include <container.h>
```

Inheritance diagram for `GiNaC::container_storage< C >`:



## Protected Types

- typedef C< [ex](#) > [STLT](#)

## Protected Member Functions

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Static Protected Member Functions

- static void [reserve](#) ([STLT](#) &, size\_t)

## Protected Attributes

- [STLT seq](#)

## 8.25.1 Detailed Description

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container_storage< C >
```

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

## 8.25.2 Member Typedef Documentation

### 8.25.2.1 STLT

```
template<template< class T, class=std::allocator< T > > class C>
typedef C<ex> GiNaC::container_storage< C >::STLT [protected]
```

## 8.25.3 Constructor & Destructor Documentation

### 8.25.3.1 container\_storage() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container\_storage< C >::container_storage () [inline], [protected]
```

Referenced by [GiNaC::container< std::vector >::container\(\)](#), and [GiNaC::container< std::vector >::container\(\)](#).

**8.25.3.2 container\_storage() [2/4]**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
    size_t n,
    const ex & e) [inline], [protected]
```

**8.25.3.3 container\_storage() [3/4]**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
    std::initializer_list< ex > il) [inline], [protected]
```

**8.25.3.4 container\_storage() [4/4]**

```
template<template< class T, class=std::allocator< T > > class C>
template<class In>
GiNaC::container_storage< C >::container_storage (
    In b,
    In e) [inline], [protected]
```

**8.25.3.5 ~container\_storage()**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::~~container_storage () [inline], [protected]
```

**8.25.4 Member Function Documentation****8.25.4.1 reserve() [1/4]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container_storage< C >::reserve (
    size_t ) [inline], [protected]
```

Referenced by [GiNaC::container< std::vector >::conjugate\(\)](#), [GiNaC::container< std::vector >::imag\\_part\(\)](#), [GiNaC::container< std::vector >::read\\_archive\(\)](#), [GiNaC::container< std::vector >::real\\_part\(\)](#), and [GiNaC::container< C >::subs](#)

**8.25.4.2 reserve() [2/4]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container_storage< C >::reserve (
    STLT & ,
    size_t ) [inline], [static], [protected]
```

**8.25.4.3 reserve() [3/4]**

```
void GiNaC::container_storage< std::vector >::reserve (
    size_t n) [inline], [protected]
```

#### 8.25.4.4 `reserve()` [4/4]

```
void GiNaC::container_storage< std::vector >::reserve (
    std::vector< ex > & v,
    size_t n) [inline], [protected]
```

### 8.25.5 Member Data Documentation

#### 8.25.5.1 `seq`

```
template<template< class T, class=std::allocator< T > > class C>
STLT GiNaC::container_storage< C >::seq [protected]
```

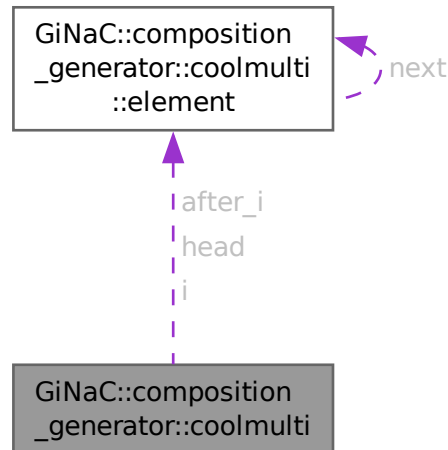
Referenced by `GiNaC::indexed::all_index_values_are()`, `GiNaC::container< C >::append()`, `GiNaC::container< std::vector >::archive()`, `GiNaC::container< std::vector >::begin()`, `GiNaC::ncmul::coeff()`, `GiNaC::container< std::vector >::conjugate()`, `GiNaC::function::conjugate()`, `GiNaC::container< std::vector >::container()`, `GiNaC::container< std::vector >::container()`, `GiNaC::ncmul::degree()`, `GiNaC::fderivative::derivative()`, `GiNaC::function::derivative()`, `GiNaC::ncmul::derivative()`, `GiNaC::clifford::do_print_dflt()`, `GiNaC::clifford::do_print_latex()`, `GiNaC::clifford::do_print_tree()`, `GiNaC::container< C >::do_print_tree()`, `GiNaC::fderivative::do_print_tree()`, `GiNaC::indexed::do_print_tree()`, `GiNaC::container< std::vector >::end()`, `GiNaC::fderivative::eval()`, `GiNaC::function::eval()`, `GiNaC::indexed::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::function::eval_ncmul()`, `GiNaC::function::evalf()`, `GiNaC::ncmul::evalm()`, `GiNaC::function::expand()`, `GiNaC::indexed::expand()`, `GiNaC::ncmul::expand()`, `GiNaC::ncmul::expandchildren()`, `GiNaC::function::expl_derivative()`, `GiNaC::indexed::get_dummy_indices()`, `GiNaC::ncmul::get_factors()`, `GiNaC::indexed::get_free_indices()`, `GiNaC::indexed::get_indices()`, `GiNaC::indexed::has_dummy_index()`, `GiNaC::function::imag_part()`, `GiNaC::indexed::indexed()`, `GiNaC::indexed::indexed()`, `GiNaC::function::info()`, `GiNaC::indexed::info()`, `GiNaC::remember_table_entry::is_equal()`, `GiNaC::container< C >::is_equal_same_type()`, `GiNaC::ncmul::ldegree()`, `GiNaC::container< C >::let_op()`, `GiNaC::container< std::vector >::nops()`, `GiNaC::container< C >::op()`, `GiNaC::function::pderivative()`, `GiNaC::function::power()`, `GiNaC::container< C >::prepend()`, `GiNaC::function::print()`, `GiNaC::indexed::print_indexed()`, `GiNaC::indexed::printindices()`, `GiNaC::container< C >::printseq()`, `GiNaC::container< std::vector >::read_archive()`, `GiNaC::function::read_archive()`, `GiNaC::indexed::read_archive()`, `GiNaC::function::real_part()`, `GiNaC::container< C >::remove_all()`, `GiNaC::container< C >::remove_first()`, `GiNaC::container< C >::remove_last()`, `GiNaC::container< std::vector >::rend()`, `GiNaC::indexed::reposition_dummy_indices()`, `GiNaC::function::return_type()`, `GiNaC::ncmul::return_type()`, `GiNaC::function::return_type_tinfo()`, `GiNaC::ncmul::return_type_tinfo()`, `GiNaC::function::series()`, `GiNaC::indexed::simplify_indexed()`, `GiNaC::container< std::vector >::sort_()`, `GiNaC::container< std::vector >::subschildren()`, `GiNaC::container< std::vector >::unique_()`, and `GiNaC::indexed::validate()`.

The documentation for this class was generated from the following file:

- [container.h](#)

## 8.26 GiNaC::composition\_generator::coolmulti Struct Reference

Collaboration diagram for GiNaC::composition\_generator::coolmulti:



### Classes

- struct [element](#)

### Public Member Functions

- [coolmulti](#) (const std::vector< unsigned > &partition)
- [~coolmulti](#) ()
- void [next\\_permutation](#) ()
- bool [finished](#) () const

### Public Attributes

- [element](#) \* [head](#)
- [element](#) \* [i](#)
- [element](#) \* [after\\_i](#)

### 8.26.1 Constructor & Destructor Documentation

#### 8.26.1.1 coolmulti()

```

GiNaC::composition_generator::coolmulti::coolmulti (
    const std::vector< unsigned > & partition) [inline], [explicit]
  
```

References [after\\_i](#), [head](#), and [i](#).

### 8.26.1.2 `~coolmulti()`

```
GiNaC::composition_generator::coolmulti::~~coolmulti () [inline]
```

References [head](#).

## 8.26.2 Member Function Documentation

### 8.26.2.1 `next_permutation()`

```
void GiNaC::composition_generator::coolmulti::next_permutation () [inline]
```

References [after\\_i](#), [head](#), [i](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::e](#).

### 8.26.2.2 `finished()`

```
bool GiNaC::composition_generator::coolmulti::finished () const [inline]
```

References [after\\_i](#), and [head](#).

## 8.26.3 Member Data Documentation

### 8.26.3.1 `head`

```
element* GiNaC::composition_generator::coolmulti::head
```

Referenced by [coolmulti\(\)](#), [finished\(\)](#), [next\\_permutation\(\)](#), and [~coolmulti\(\)](#).

### 8.26.3.2 `i`

```
element * GiNaC::composition_generator::coolmulti::i
```

Referenced by [coolmulti\(\)](#), and [next\\_permutation\(\)](#).

### 8.26.3.3 `after_i`

```
element * GiNaC::composition_generator::coolmulti::after_i
```

Referenced by [coolmulti\(\)](#), [finished\(\)](#), and [next\\_permutation\(\)](#).

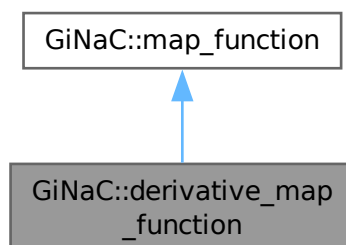
The documentation for this struct was generated from the following file:

- [utils.h](#)

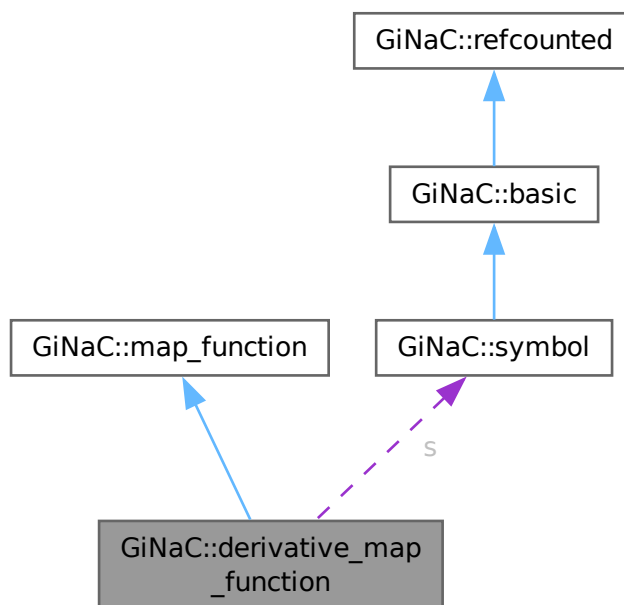
## 8.27 GiNaC::derivative\_map\_function Struct Reference

Function object to be applied by [basic::derivative\(\)](#).

Inheritance diagram for GiNaC::derivative\_map\_function:



Collaboration diagram for GiNaC::derivative\_map\_function:



### Public Member Functions

- [derivative\\_map\\_function](#) (const [symbol](#) &sym)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Public Attributes

- const [symbol](#) & [s](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.27.1 Detailed Description

Function object to be applied by [basic::derivative\(\)](#).

### 8.27.2 Constructor & Destructor Documentation

#### 8.27.2.1 [derivative\\_map\\_function\(\)](#)

```
GiNaC::derivative_map_function::derivative_map_function (
    const symbol & sym) [inline]
```

References [s](#).

### 8.27.3 Member Function Documentation

#### 8.27.3.1 [operator>\(\)\(\)](#)

```
ex GiNaC::derivative_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::diff\(\)](#), and [s](#).

### 8.27.4 Member Data Documentation

#### 8.27.4.1 [s](#)

```
const symbol& GiNaC::derivative_map_function::s
```

Referenced by [derivative\\_map\\_function\(\)](#), and [operator>\(\)\(\)](#).

The documentation for this struct was generated from the following file:

- [basic.cpp](#)



## 8.28 GiNaC::determinant\_algo Class Reference

Switch to control algorithm for determinant computation.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#) , [gauss](#) , [divfree](#) , [laplace](#) ,  
[bareiss](#) }

### 8.28.1 Detailed Description

Switch to control algorithm for determinant computation.

### 8.28.2 Member Enumeration Documentation

#### 8.28.2.1 anonymous enum

anonymous enum

#### Enumerator

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>The determinant is then just the product of diagonal elements. Choose this algorithm only for purely numerical matrices.</p>
divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>The determinant can later be computed by inspecting the diagonal elements only. This algorithm is only there for the purpose of cross-checks. It is never fast.</p>
laplace	Laplace elimination. This is plain recursive elimination along minors although multiple minors are avoided by the algorithm. Although the algorithm is exponential in complexity it is frequently the fastest one when the matrix is populated by complicated symbolic expressions.

bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)}m_{k,k}^{(k)} - m_{i,k}^{(k)}m_{k,j}^{(k)})/m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. The determinant can then be read of from the lower right entry. This algorithm is rarely fast for computing determinants.</p>
---------	--

The documentation for this class was generated from the following file:

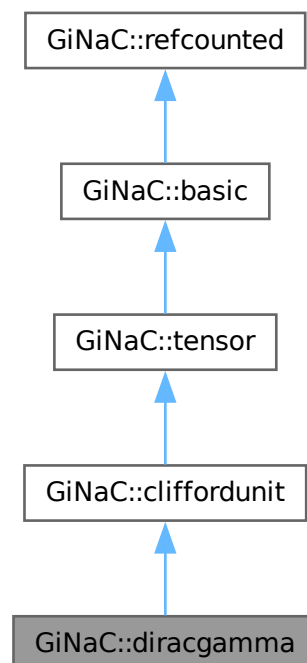
- [flags.h](#)

## 8.29 GiNaC::diracgamma Class Reference

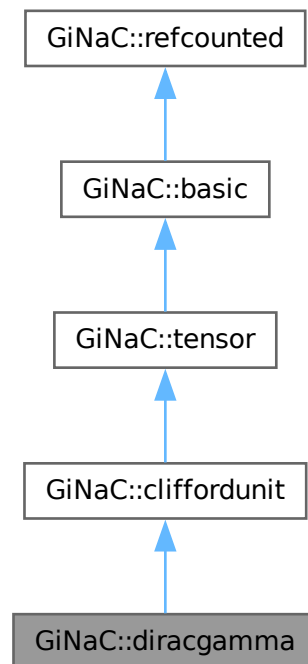
This class represents the Dirac gamma Lorentz vector.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma:



Collaboration diagram for GiNaC::diracgamma:



### Public Member Functions

- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of a gamma matrix with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const basic &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const

- Perform automatic non-interruptive term rewriting rules.*

  - virtual `ex evalf` () const

*Evaluate object numerically.*
- virtual `ex evalm` () const

*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const

*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const

*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprnttree` () const

*Little wrapper around prnttree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual size\_t `nops` () const

*Number of operands/members.*
- virtual `ex op` (size\_t i) const

*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const

*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const

- *Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- *Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const
- *Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- *Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- *Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- *Multiply an indexed expression with a scalar.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const
- *Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::cliffordunit`

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 8.29.1 Detailed Description

This class represents the Dirac gamma Lorentz vector.

## 8.29.2 Member Function Documentation

### 8.29.2.1 contract\_with()

```
bool GiNaC::diracgamma::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of a gamma matrix with something else.

Reimplemented from [GiNaC::cliffordunit](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex\\_2](#), [GiNaC::base\\_and\\_index\(\)](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::dirac\\_slash\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::lorentz\\_g\(\)](#), and [GiNaC::minimal\\_dim\(\)](#).

### 8.29.2.2 do\_print()

```
void GiNaC::diracgamma::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.29.2.3 do\_print\_latex()

```
void GiNaC::diracgamma::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

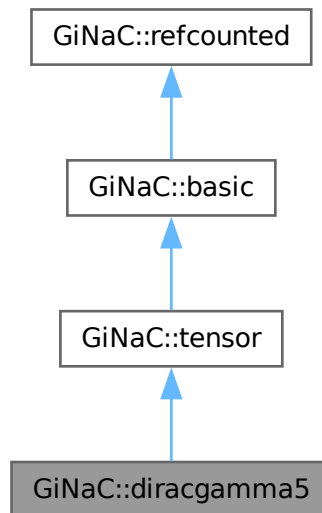
- [clifford.h](#)
- [clifford.cpp](#)

## 8.30 GiNaC::diracgamma5 Class Reference

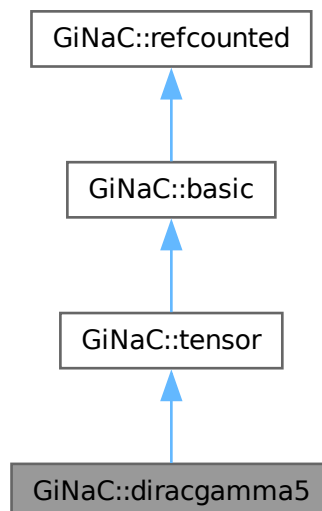
This class represents the Dirac gamma5 object which anticommutes with all other gammas.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma5:



Collaboration diagram for GiNaC::diracgamma5:





**Protected Member Functions**

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

**Protected Member Functions inherited from `GiNaC::tensor`**

- unsigned `return_type` () const override

**Protected Member Functions inherited from `GiNaC::basic`**

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Private Member Functions**

- `ex conjugate` () const override

**Additional Inherited Members****Public Member Functions inherited from `GiNaC::tensor`**

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.30.1 Detailed Description

This class represents the Dirac gamma5 object which anticommutes with all other gammas.

### 8.30.2 Member Function Documentation

#### 8.30.2.1 [conjugate\(\)](#)

```
ex GiNaC::diracgamma5::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex\\_1](#).

#### 8.30.2.2 [do\\_print\(\)](#)

```
void GiNaC::diracgamma5::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

#### 8.30.2.3 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracgamma5::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

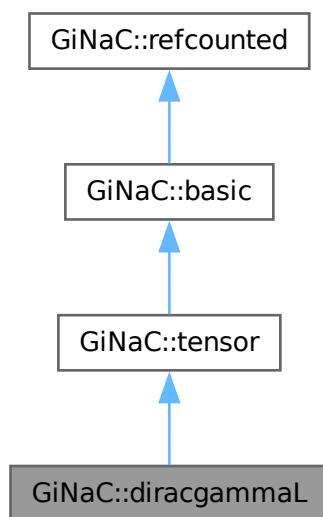
- [clifford.h](#)
- [clifford.cpp](#)

## 8.31 GiNaC::diracgammaL Class Reference

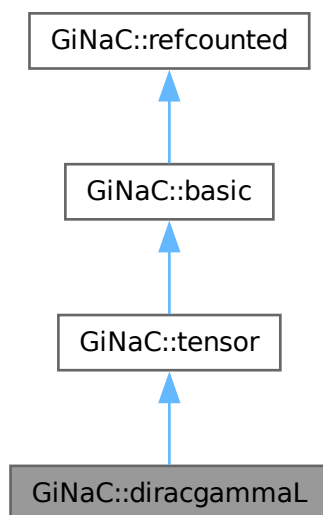
This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaL:



Collaboration diagram for GiNaC::diracgammaL:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Private Member Functions

- `ex conjugate` () const override

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned options) const

*Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*

  - const `basic` & `hold` () const

*Stop further evaluation.*



- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.31.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

## 8.31.2 Member Function Documentation

### 8.31.2.1 conjugate()

```
ex GiNaC::diracgammaL::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#).

### 8.31.2.2 do\_print()

```
void GiNaC::diracgammaL::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

### 8.31.2.3 do\_print\_latex()

```
void GiNaC::diracgammaL::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

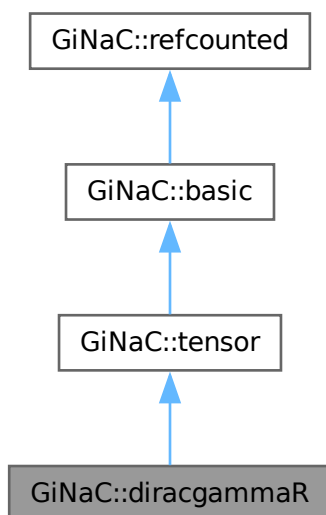
- [clifford.h](#)
- [clifford.cpp](#)

## 8.32 GiNaC::diracgammaR Class Reference

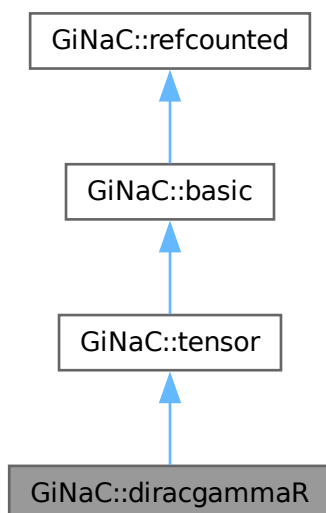
This class represents the Dirac gammaL object which behaves like  $\frac{1}{2} (1+\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaR:



Collaboration diagram for GiNaC::diracgammaR:



**Protected Member Functions**

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

**Protected Member Functions inherited from `GiNaC::tensor`**

- unsigned `return_type` () const override

**Protected Member Functions inherited from `GiNaC::basic`**

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Private Member Functions**

- `ex conjugate` () const override

**Additional Inherited Members****Public Member Functions inherited from `GiNaC::tensor`**

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.32.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $\frac{1}{2} (1 + \gamma_5)$ .

### 8.32.2 Member Function Documentation

#### 8.32.2.1 [conjugate\(\)](#)

```
ex GiNaC::diracgammaR::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#).

#### 8.32.2.2 [do\\_print\(\)](#)

```
void GiNaC::diracgammaR::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

#### 8.32.2.3 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracgammaR::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

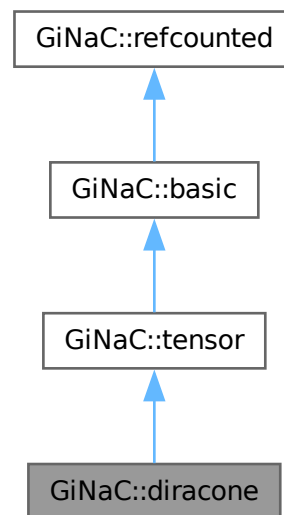
- [clifford.h](#)
- [clifford.cpp](#)

## 8.33 GiNaC::diracone Class Reference

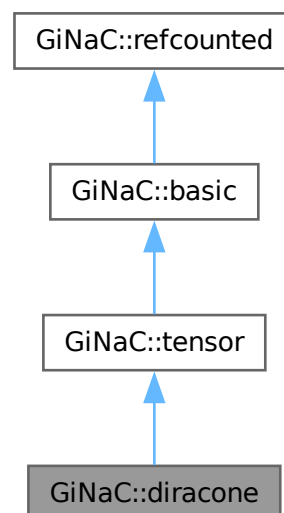
This class represents the Clifford algebra unity element.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracone:



Collaboration diagram for GiNaC::diracone:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*



## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const

*Set some [status\\_flags](#).*

- const [basic](#) & [clearflag](#) (unsigned f) const

*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.33.1 Detailed Description

This class represents the Clifford algebra unity element.

### 8.33.2 Member Function Documentation

#### 8.33.2.1 [do\\_print\(\)](#)

```
void GiNaC::diracone::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

#### 8.33.2.2 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracone::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following file:

- [clifford.h](#)

## 8.34 GiNaC::do\_taylor Class Reference

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

```
#include <function.h>
```

### 8.34.1 Detailed Description

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

The documentation for this class was generated from the following file:

- [function.h](#)

## 8.35 GiNaC::domain Class Reference

Domain of an object.

```
#include <flags.h>
```

### Public Types

- enum { [complex](#) , [real](#) , [positive](#) }

### 8.35.1 Detailed Description

Domain of an object.

### 8.35.2 Member Enumeration Documentation

#### 8.35.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

complex	
real	
positive	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.36 GiNaC::dunno Class Reference

Exception class thrown by functions to signal unimplemented functionality so the expression may just be `.hold()`.

```
#include <utils.h>
```

### 8.36.1 Detailed Description

Exception class thrown by functions to signal unimplemented functionality so the expression may just be `.hold()`.

The documentation for this class was generated from the following file:

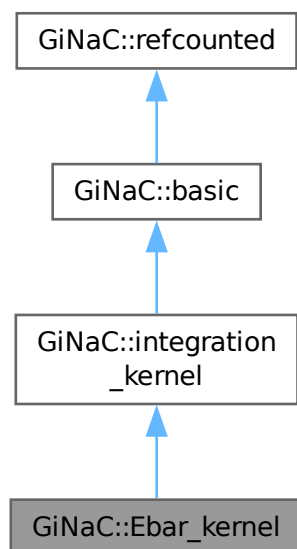
- [utils.h](#)

## 8.37 GiNaC::Ebar\_kernel Class Reference

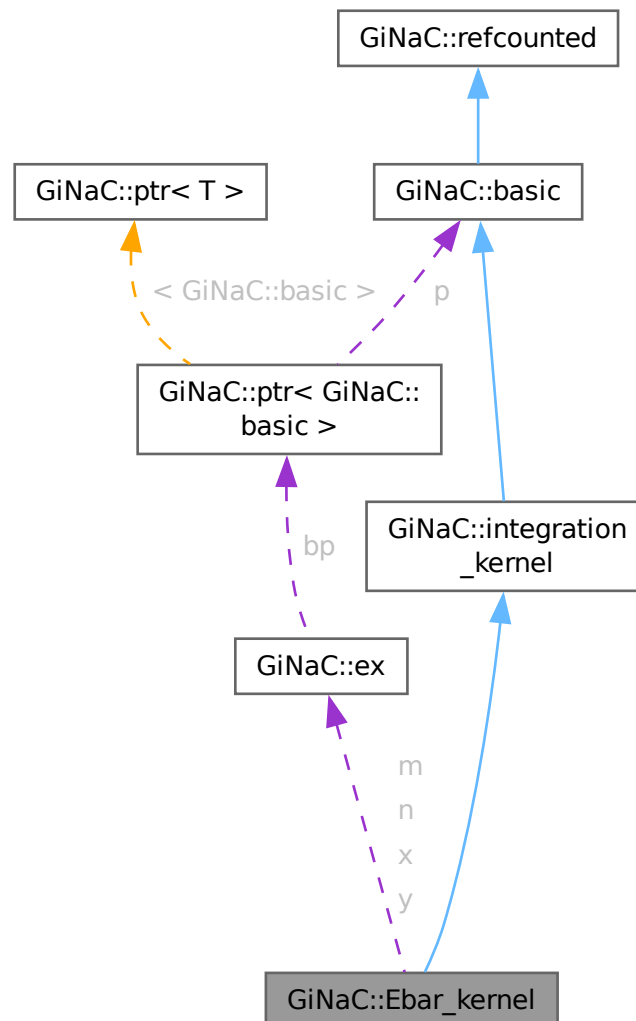
The Ebar-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Ebar\_kernel:



Collaboration diagram for GiNaC::Ebar\_kernel:



## Public Member Functions

- `Ebar_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of `Ebar_{n,m}(x,y,qbar)`.*

## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)



- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Load (deserialize) the object from an archive node.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Helper function for `subs()`.*
- `int compare` (const `basic` &other) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `bool is_equal` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- [std::vector< cln::cl\\_N >](#) [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.37.1 Detailed Description

The Ebar-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\bar{E}}(x; y) = \bar{E}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

## 8.37.2 Constructor & Destructor Documentation

### 8.37.2.1 Ebar\_kernel()

```
GiNaC::Ebar_kernel::Ebar_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y)
```

References [Ebar\\_kernel\(\)](#), [m](#), [n](#), [x](#), and [y](#).

Referenced by [Ebar\\_kernel\(\)](#).

## 8.37.3 Member Function Documentation

### 8.37.3.1 nops()

```
size_t GiNaC::Ebar_kernel::nops () const \[override\], \[virtual\]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.37.3.2 op()

```
ex GiNaC::Ebar_kernel::op (
    size_t i) const \[override\], \[virtual\]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 8.37.3.3 let\_op()

```
ex & GiNaC::Ebar_kernel::let_op (
    size_t i) \[override\], \[virtual\]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

#### 8.37.3.4 is\_numeric()

```
bool GiNaC::Ebar_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

#### 8.37.3.5 get\_numerical\_value()

```
ex GiNaC::Ebar_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

#### 8.37.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Ebar_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex\\_to\(\)](#), [m](#), [n](#), [x](#), and [y](#).

#### 8.37.3.7 do\_print()

```
void GiNaC::Ebar_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [m](#), [n](#), [GiNaC::print\\_context::s](#), [x](#), and [y](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.37.4 Member Data Documentation

#### 8.37.4.1 n

`ex` GiNaC::Ebar\_kernel::n [protected]

Referenced by [do\\_print\(\)](#), [Ebar\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.37.4.2 m

`ex` GiNaC::Ebar\_kernel::m [protected]

Referenced by [do\\_print\(\)](#), [Ebar\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.37.4.3 x

`ex` GiNaC::Ebar\_kernel::x [protected]

Referenced by [do\\_print\(\)](#), [Ebar\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.37.4.4 y

`ex` GiNaC::Ebar\_kernel::y [protected]

Referenced by [do\\_print\(\)](#), [Ebar\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

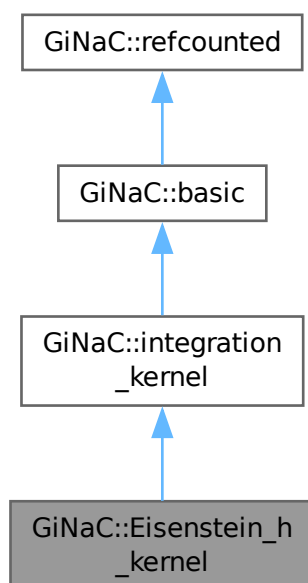
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.38 GiNaC::Eisenstein\_h\_kernel Class Reference

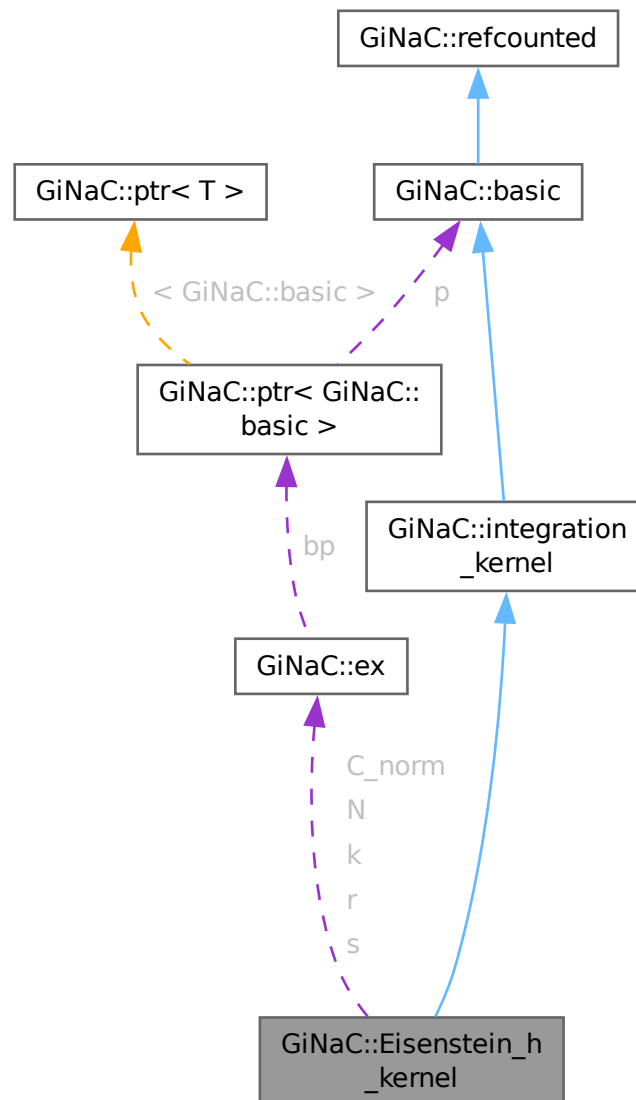
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_h\_kernel:



Collaboration diagram for GiNaC::Eisenstein\_h\_kernel:



## Public Member Functions

- `Eisenstein_h_kernel` (const `ex` &`k`, const `ex` &`N`, const `ex` &`r`, const `ex` &`s`, const `ex` &`C_norm=numeric(1)`)
- `ex series` (const `relational` &`r`, int order, unsigned options=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex` & `let_op` (size\_t `i`) override

- *Return modifiable operand/member at position i.*
- `bool is_numeric (void) const` override
  - This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series (const ex &x, int order) const` override
  - Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value (const ex &qbar, int N_trunc=0) const` override
  - Returns the value of the modular form.*
- `ex coefficient_a0 (const numeric &k, const numeric &r, const numeric &s, const numeric &N) const`
  - The constant coefficient in the Fourier expansion.*
- `ex coefficient_an (const numeric &n, const numeric &k, const numeric &r, const numeric &s, const numeric &N) const`
  - The higher coefficients in the Fourier expansion.*
- `ex q_expansion_modular_form (const ex &q, int order) const`

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series (const relational &r, int order, unsigned options=0) const` override
  - Default implementation of `ex::series()`.*
- `virtual bool has_trailing_zero (void) const`
  - This routine returns true, if the integration kernel has a trailing zero.*
- `size_t get_cache_size (void) const`
  - Returns the current size of the cache.*
- `void set_cache_step (int cache_steps) const`
  - Sets the step size by which the cache is increased.*
- `ex get_series_coeff (int i) const`
  - Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff (int i) const`
  - Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`
  - basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`
  - basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`
  - Create a clone of this object on the heap.*
- `virtual ex eval () const`
  - Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf () const`
  - Evaluate object numerically.*
- `virtual ex evalm () const`
  - Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ () const`
  - Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`
  - Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`



- virtual void [dbgprint](#) () const  
*Output to stream.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) (GiNaC::visitor &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const

*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- bool [uses\\_Laurent\\_series](#) () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method [Laurent\\_series](#) needs to be implemented).*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual [cln::cl\\_N](#) [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [ex](#) [get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex k](#)
- [ex N](#)
- [ex r](#)
- [ex s](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.38.1 Detailed Description

The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,r,s}^{\text{Eisenstein,h}} = C_k h_{k,N,r,s}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

## 8.38.2 Constructor & Destructor Documentation

### 8.38.2.1 Eisenstein\_h\_kernel()

```
GiNaC::Eisenstein_h_kernel::Eisenstein_h_kernel (
    const ex & k,
    const ex & N,
    const ex & r,
    const ex & s,
    const ex & C_norm = numeric(1))
```

References [C\\_norm](#), [Eisenstein\\_h\\_kernel\(\)](#), [k](#), [N](#), [r](#), and [s](#).

Referenced by [Eisenstein\\_h\\_kernel\(\)](#).

## 8.38.3 Member Function Documentation

### 8.38.3.1 series()

```
ex GiNaC::Eisenstein_h_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0) const \[override\], \[virtual\]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::basic](#).

References [q\\_expansion\\_modular\\_form\(\)](#), [r](#), and [GiNaC::ex::series\(\)](#).

### 8.38.3.2 nops()

```
size_t GiNaC::Eisenstein_h_kernel::nops () const \[override\], \[virtual\]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.38.3.3 op()

```
ex GiNaC::Eisenstein_h_kernel::op (
    size_t i) const \[override\], \[virtual\]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), [N](#), [r](#), and [s](#).

#### 8.38.3.4 let\_op()

```
ex & GiNaC::Eisenstein_h_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), [N](#), [r](#), and [s](#).

#### 8.38.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_h_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::info\\_flags::integer](#), [k](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [r](#), and [s](#).

#### 8.38.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_h_kernel::Laurent_series (
    const ex & x,
    int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [q\\_expansion\\_modular\\_form\(\)](#), and [GiNaC::ex::series\(\)](#).

#### 8.38.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_h_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), and [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#).

**8.38.3.8 uses\_Laurent\_series()**

```
bool GiNaC::Eisenstein_h_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

**8.38.3.9 coefficient\_a0()**

```
ex GiNaC::Eisenstein_h_kernel::coefficient_a0 (
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N) const
```

The constant coefficient in the Fourier expansion.

References [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::I](#), [GiNaC::irem\(\)](#), [k](#), [GiNaC::mod\(\)](#), [N](#), [GiNaC::Pi](#), [r](#), [s](#), and [GiNaC::sin\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

**8.38.3.10 coefficient\_an()**

```
ex GiNaC::Eisenstein_h_kernel::coefficient_an (
    const numeric & n,
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N) const
```

The higher coefficients in the Fourier expansion.

References [GiNaC::exp\(\)](#), [GiNaC::I](#), [GiNaC::irem\(\)](#), [k](#), [GiNaC::mod\(\)](#), [N](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [r](#), and [s](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

**8.38.3.11 q\_expansion\_modular\_form()**

```
ex GiNaC::Eisenstein_h_kernel::q_expansion_modular_form (
    const ex & q,
    int order) const
```

References [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [GiNaC::ex\\_to\(\)](#), [k](#), [N](#), [GiNaC::pow\(\)](#), [r](#), [s](#), and [GiNaC::ex::series\(\)](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 8.38.3.12 do\_print()

```
void GiNaC::Eisenstein_h_kernel::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [C\\_norm](#), [k](#), [N](#), [r](#), [s](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.38.4 Member Data Documentation

### 8.38.4.1 k

```
ex GiNaC::Eisenstein_h_kernel::k [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [Eisenstein\\_h\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 8.38.4.2 N

```
ex GiNaC::Eisenstein_h_kernel::N [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [Eisenstein\\_h\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 8.38.4.3 r

```
ex GiNaC::Eisenstein_h_kernel::r [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [Eisenstein\\_h\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [q\\_expansion\\_modular\\_form\(\)](#), and [series\(\)](#).

### 8.38.4.4 s

```
ex GiNaC::Eisenstein_h_kernel::s [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [Eisenstein\\_h\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 8.38.4.5 C\_norm

```
ex GiNaC::Eisenstein_h_kernel::C_norm [protected]
```

Referenced by [do\\_print\(\)](#), [Eisenstein\\_h\\_kernel\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

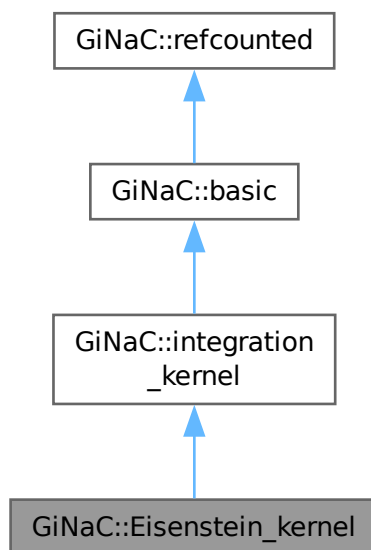
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

### 8.39 GiNaC::Eisenstein\_kernel Class Reference

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

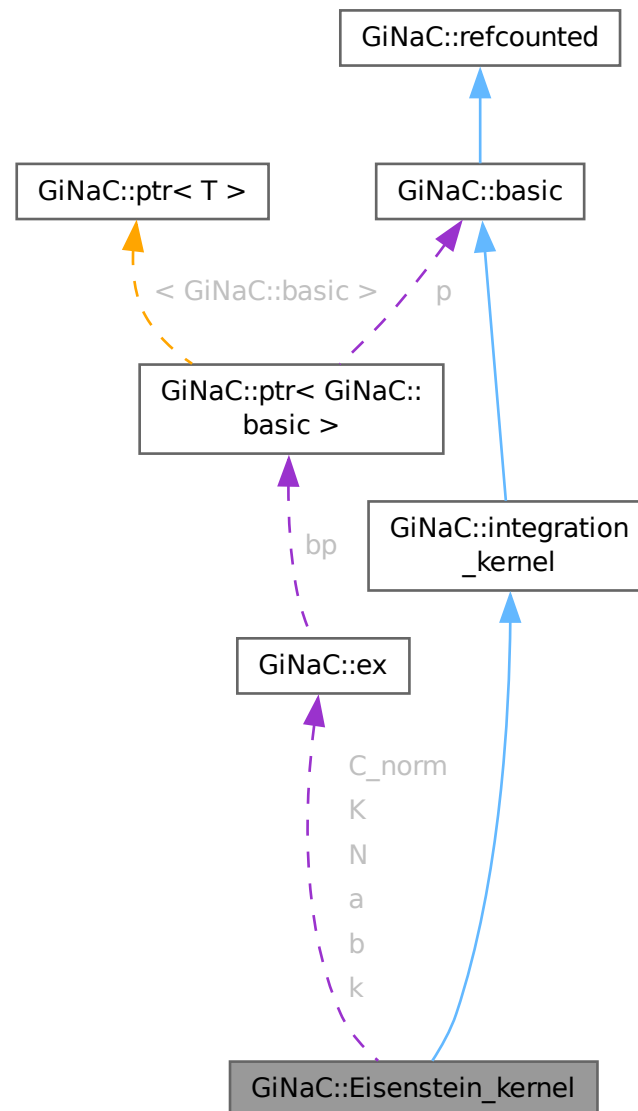
```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_kernel:





Collaboration diagram for GiNaC::Eisenstein\_kernel:



## Public Member Functions

- `Eisenstein_kernel` (const `ex` &k, const `ex` &N, const `ex` &a, const `ex` &b, const `ex` &K, const `ex` &C\_norm=numeric(1))
- `ex_series` (const `relational` &r, int order, unsigned options=0) const override
 

*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override
 

*Number of operands/members.*
- `ex op` (size\_t i) const override
 

*Return operand/member at position i.*

- `ex` & `let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- bool `is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex` `Laurent_series` (const `ex` &x, int order) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex` `get_numerical_value` (const `ex` &qbar, int N\_trunc=0) const override  
*Returns the value of the modular form.*
- `ex` `q_expansion_modular_form` (const `ex` &q, int order) const

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex` `series` (const `relational` &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex` `get_series_coeff` (int i) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N` `series_coeff` (int i) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex` `eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex` `evalf` () const  
*Evaluate object numerically.*
- virtual `ex` `evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex` `eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex` `eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const

- Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const
- Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const
- Information about the object.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const
- Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const
- Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const
- Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const
- Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const
- Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const
- Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const
- Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const
- Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const
- Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const
- Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const
- Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const
- Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const
- Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const
- Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const

- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex k](#)
- [ex N](#)
- [ex a](#)
- [ex b](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- [std::vector< cln::cl\\_N >](#) [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.39.1 Detailed Description

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,a,b,K}^{\text{Eisenstein}} = C_k E_{k,N,a,b,K}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

The integers a and b are either one or the discriminant of a quadratic number field. This class represents Eisenstein series, which can be defined by primitive Dirichlet characters from the Kronecker symbol. This implies that the characters take the values -1,0,1, i.e. no higher roots of unity occur. The

$$\bar{q}$$

-expansion has then rational coefficients.

Ref.: W. Stein, Modular Forms: A Computational Approach, Chapter 5

### 8.39.2 Constructor & Destructor Documentation

#### 8.39.2.1 Eisenstein\_kernel()

```
GiNaC::Eisenstein_kernel::Eisenstein_kernel (
    const ex & k,
    const ex & N,
    const ex & a,
    const ex & b,
    const ex & K,
    const ex & C_norm = numeric(1))
```

References [a](#), [b](#), [C\\_norm](#), [Eisenstein\\_kernel\(\)](#), [K](#), [k](#), and [N](#).

Referenced by [Eisenstein\\_kernel\(\)](#).

### 8.39.3 Member Function Documentation

#### 8.39.3.1 series()

```
ex GiNaC::Eisenstein_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::relational::lhs\(\)](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::relational::rhs\(\)](#), and [GiNaC::ex::series\(\)](#).

### 8.39.3.2 nops()

```
size_t GiNaC::Eisenstein_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.39.3.3 op()

```
ex GiNaC::Eisenstein_kernel::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [K](#), [k](#), and [N](#).

### 8.39.3.4 let\_op()

```
ex & GiNaC::Eisenstein_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [k](#), and [N](#).

### 8.39.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::info\\_flags::integer](#), [K](#), [k](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::info\\_flags::posint](#).

### 8.39.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_kernel::Laurent_series (
    const ex & x,
    int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [q\\_expansion\\_modular\\_form\(\)](#), and [GiNaC::ex::series\(\)](#).

### 8.39.3.7 `get_numerical_value()`

```
ex GiNaC::Eisenstein_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), and [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#).

### 8.39.3.8 `uses_Laurent_series()`

```
bool GiNaC::Eisenstein_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 8.39.3.9 `q_expansion_modular_form()`

```
ex GiNaC::Eisenstein_kernel::q_expansion_modular_form (
    const ex & q,
    int order) const
```

References [a](#), [b](#), [GiNaC::ex\\_to\(\)](#), [K](#), [k](#), and [N](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 8.39.3.10 `do_print()`

```
void GiNaC::Eisenstein_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [a](#), [b](#), [C\\_norm](#), [K](#), [k](#), [N](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.39.4 Member Data Documentation

### 8.39.4.1 `k`

```
ex GiNaC::Eisenstein_kernel::k [protected]
```

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).



**8.39.4.2 N**

`ex GiNaC::Eisenstein_kernel::N [protected]`

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**8.39.4.3 a**

`ex GiNaC::Eisenstein_kernel::a [protected]`

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**8.39.4.4 b**

`ex GiNaC::Eisenstein_kernel::b [protected]`

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**8.39.4.5 K**

`ex GiNaC::Eisenstein_kernel::K [protected]`

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**8.39.4.6 C\_norm**

`ex GiNaC::Eisenstein_kernel::C_norm [protected]`

Referenced by [do\\_print\(\)](#), [Eisenstein\\_kernel\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

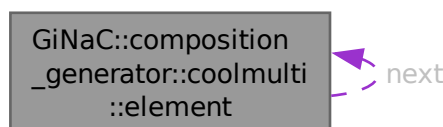
The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.40 GiNaC::composition\_generator::coolmulti::element Struct Reference

```
#include <utils.h>
```

Collaboration diagram for GiNaC::composition\_generator::coolmulti::element:



## Public Member Functions

- [element](#) (unsigned val, element \*n)
- [~element](#) ()

## Public Attributes

- unsigned [value](#)
- [element](#) \* [next](#)

## 8.40.1 Constructor & Destructor Documentation

### 8.40.1.1 [element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::element (
    unsigned val,
    element * n) [inline]
```

References [element\(\)](#), [next](#), and [value](#).

Referenced by [element\(\)](#).

### 8.40.1.2 [~element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::~~element () [inline]
```

References [next](#).

## 8.40.2 Member Data Documentation

### 8.40.2.1 [value](#)

```
unsigned GiNaC::composition_generator::coolmulti::element::value
```

Referenced by [element\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), and [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#).

### 8.40.2.2 [next](#)

```
element* GiNaC::composition_generator::coolmulti::element::next
```

Referenced by [element\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#), and [~element\(\)](#).

The documentation for this struct was generated from the following file:

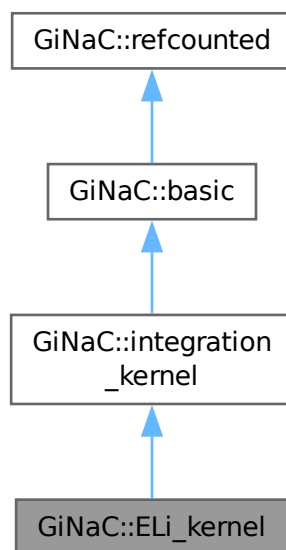
- [utils.h](#)

## 8.41 GiNaC::ELi\_kernel Class Reference

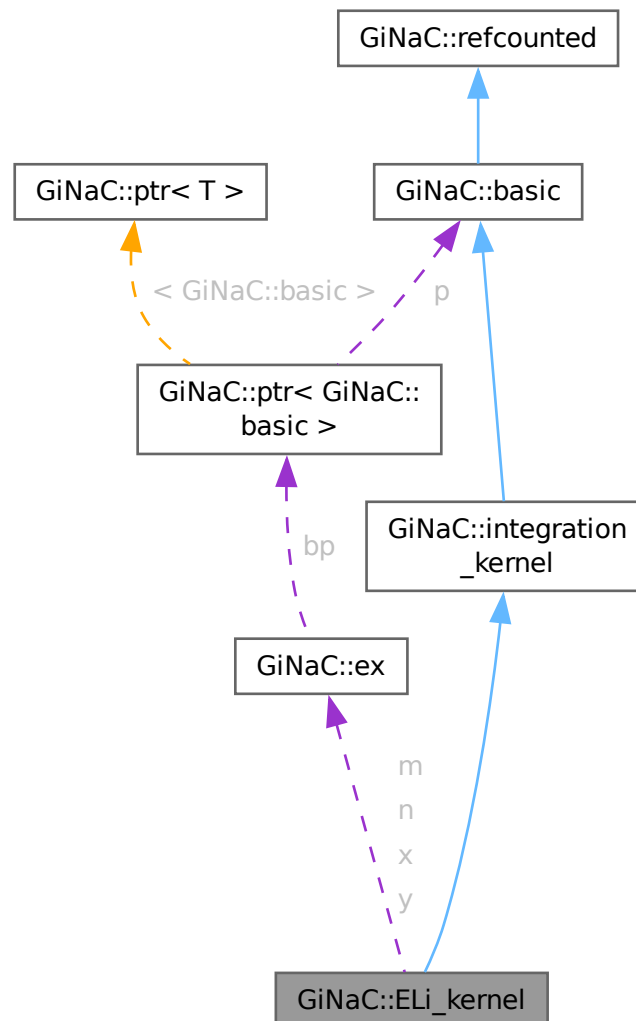
The ELi-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::ELi\_kernel:



Collaboration diagram for GiNaC::ELi\_kernel:



## Public Member Functions

- `ELi_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of  $ELi_{\{n,m\}}(x,y,qbar)$ .*

## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- [std::vector< cln::cl\\_N >](#) [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.41.1 Detailed Description

The ELi-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\text{ELi}}(x; y) = \text{ELi}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$



## 8.41.2 Constructor & Destructor Documentation

### 8.41.2.1 ELi\_kernel()

```
GiNaC::ELi_kernel::ELi_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y)
```

References [ELi\\_kernel\(\)](#), [m](#), [n](#), [x](#), and [y](#).

Referenced by [ELi\\_kernel\(\)](#).

## 8.41.3 Member Function Documentation

### 8.41.3.1 nops()

```
size_t GiNaC::ELi_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.41.3.2 op()

```
ex GiNaC::ELi_kernel::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 8.41.3.3 let\_op()

```
ex & GiNaC::ELi_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

#### 8.41.3.4 is\_numeric()

```
bool GiNaC::ELi_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

#### 8.41.3.5 get\_numerical\_value()

```
ex GiNaC::ELi_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of  $\text{ELi}_{\{n,m\}}(x,y,qbar)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#).

#### 8.41.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::ELi_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex\\_to\(\)](#), [m](#), [n](#), [x](#), and [y](#).

#### 8.41.3.7 do\_print()

```
void GiNaC::ELi_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [m](#), [n](#), [GiNaC::print\\_context::s](#), [x](#), and [y](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.41.4 Member Data Documentation

#### 8.41.4.1 n

```
ex GiNaC::ELi_kernel::n [protected]
```

Referenced by [do\\_print\(\)](#), [ELi\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.41.4.2 m

```
ex GiNaC::ELi_kernel::m [protected]
```

Referenced by [do\\_print\(\)](#), [ELi\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.41.4.3 x

```
ex GiNaC::ELi_kernel::x [protected]
```

Referenced by [do\\_print\(\)](#), [ELi\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 8.41.4.4 y

```
ex GiNaC::ELi_kernel::y [protected]
```

Referenced by [do\\_print\(\)](#), [ELi\\_kernel\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.42 std::equal\_to< GiNaC::ex > Struct Reference

Specialization of `std::equal_to()` for `ex` objects.

```
#include <ex.h>
```

### Public Member Functions

- `bool operator() (const GiNaC::ex &e1, const GiNaC::ex &e2) const` noexcept

### 8.42.1 Detailed Description

Specialization of `std::equal_to()` for `ex` objects.

### 8.42.2 Member Function Documentation

#### 8.42.2.1 operator()()

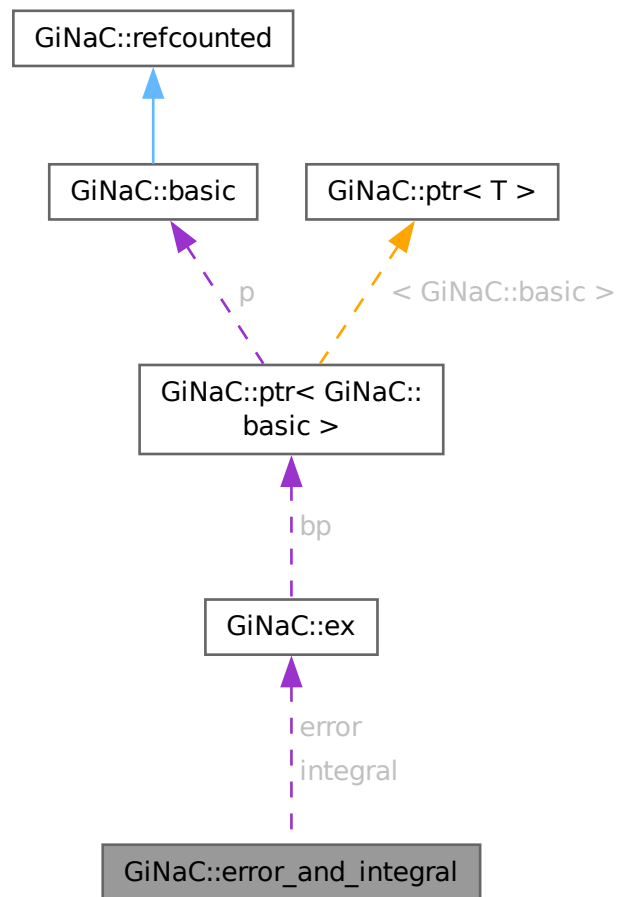
```
bool std::equal_to< GiNaC::ex >::operator() (
    const GiNaC::ex & e1,
    const GiNaC::ex & e2) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 8.43 GiNaC::error\_and\_integral Struct Reference

Collaboration diagram for GiNaC::error\_and\_integral:



### Public Member Functions

- [error\\_and\\_integral](#) (const [ex](#) &err, const [ex](#) &integ)

### Public Attributes

- [ex error](#)
- [ex integral](#)

## 8.43.1 Constructor & Destructor Documentation

### 8.43.1.1 error\_and\_integral()

```
GiNaC::error_and_integral::error_and_integral (
    const ex & err,
    const ex & integ) [inline]
```

References [error](#), and [integral](#).

## 8.43.2 Member Data Documentation

### 8.43.2.1 error

```
ex GiNaC::error_and_integral::error
```

Referenced by [error\\_and\\_integral\(\)](#), and [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)\(\)](#).

### 8.43.2.2 integral

```
ex GiNaC::error_and_integral::integral
```

Referenced by [error\\_and\\_integral\(\)](#), and [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)\(\)](#).

The documentation for this struct was generated from the following file:

- [integral.cpp](#)

## 8.44 GiNaC::error\_and\_integral\_is\_less Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [error\\_and\\_integral](#) &*e1*, const [error\\_and\\_integral](#) &*e2*) const

## 8.44.1 Member Function Documentation

### 8.44.1.1 operator()()

```
bool GiNaC::error_and_integral_is_less::operator() (
    const error\_and\_integral & e1,
    const error\_and\_integral & e2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), [GiNaC::error\\_and\\_integral::error](#), and [GiNaC::error\\_and\\_integral::integral](#).

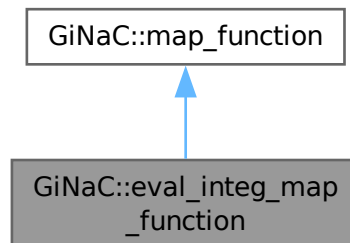
The documentation for this struct was generated from the following file:

- [integral.cpp](#)

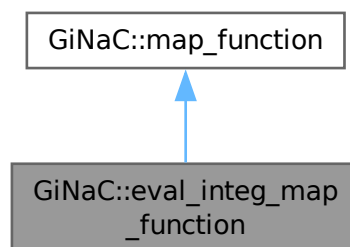
## 8.45 GiNaC::eval\_integ\_map\_function Struct Reference

Function object to be applied by [basic::eval\\_integ\(\)](#).

Inheritance diagram for GiNaC::eval\_integ\_map\_function:



Collaboration diagram for GiNaC::eval\_integ\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.45.1 Detailed Description

Function object to be applied by [basic::eval\\_integ\(\)](#).

### 8.45.2 Member Function Documentation

#### 8.45.2.1 operator()

```
ex GiNaC::eval_integ_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::eval\\_integ\(\)](#).

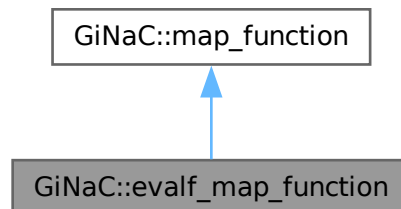
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

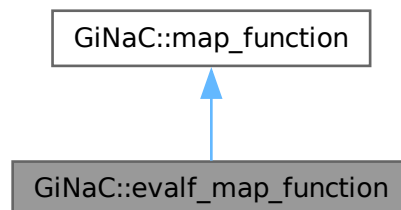
## 8.46 GiNaC::evalf\_map\_function Struct Reference

Function object to be applied by [basic::evalf\(\)](#).

Inheritance diagram for GiNaC::evalf\_map\_function:



Collaboration diagram for GiNaC::evalf\_map\_function:



## Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.46.1 Detailed Description

Function object to be applied by [basic::evalf\(\)](#).

### 8.46.2 Member Function Documentation

#### 8.46.2.1 [operator\(\)](#)()

```
ex GiNaC::evalf_map_function::operator() (  
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalf\(\)](#).

The documentation for this struct was generated from the following file:

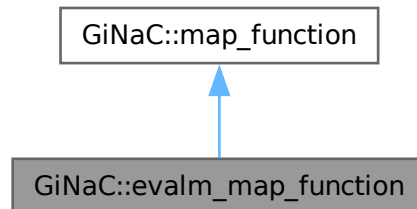
- [basic.cpp](#)



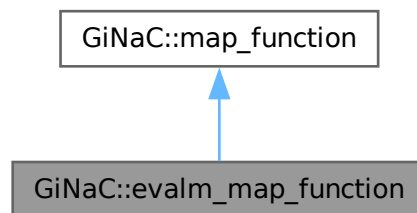
## 8.47 GiNaC::evalm\_map\_function Struct Reference

Function object to be applied by [basic::evalm\(\)](#).

Inheritance diagram for GiNaC::evalm\_map\_function:



Collaboration diagram for GiNaC::evalm\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.47.1 Detailed Description

Function object to be applied by [basic::evalm\(\)](#).

### 8.47.2 Member Function Documentation

#### 8.47.2.1 operator>()

```
ex GiNaC::evalm_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalm\(\)](#).

The documentation for this struct was generated from the following file:

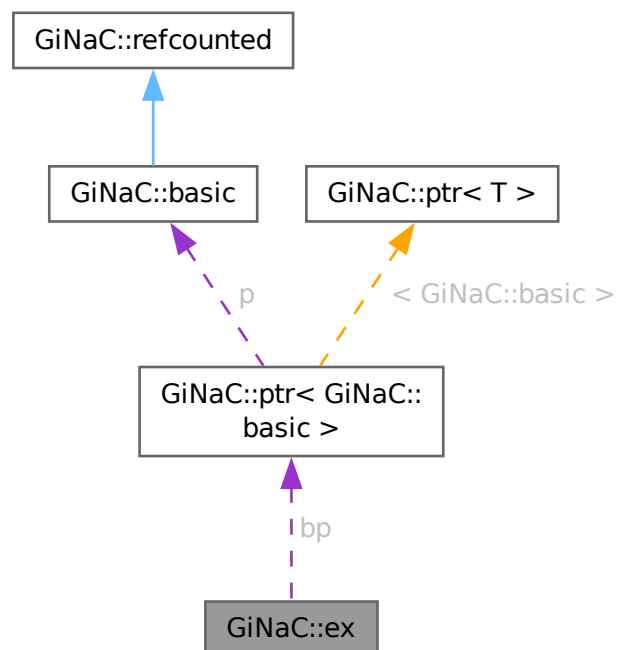
- [basic.cpp](#)

## 8.48 GiNaC::ex Class Reference

Lightweight wrapper for [GiNaC](#)'s symbolic objects.

```
#include <ex.h>
```

Collaboration diagram for [GiNaC::ex](#):



## Public Member Functions

- `ex ()` noexcept
- `ex (const basic &other)`
- `ex (int i)`
- `ex (unsigned int i)`
- `ex (long i)`
- `ex (unsigned long i)`
- `ex (long long i)`
- `ex (unsigned long long i)`
- `ex (double const d)`
- `ex (const std::string &s, const ex &l)`  
*Construct ex from string and a list of symbols.*
- `void swap (ex &other)` noexcept  
*Efficiently swap the contents of two expressions.*
- `const_iterator begin ()` const noexcept
- `const_iterator end ()` const noexcept
- `const_preorder_iterator preorder_begin ()` const
- `const_preorder_iterator preorder_end ()` const noexcept
- `const_postorder_iterator postorder_begin ()` const
- `const_postorder_iterator postorder_end ()` const noexcept
- `ex eval ()` const
- `ex evalf ()` const
- `ex evalm ()` const
- `ex eval_ncmul (const exvector &v)` const
- `ex eval_integ ()` const
- `void print (const print_context &c, unsigned level=0)` const  
*Print expression to stream.*
- `void dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- `void dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- `bool info (unsigned inf)` const
- `size_t nops ()` const
- `ex op (size_t i)` const
- `ex operator[] (const ex &index)` const
- `ex operator[] (size_t i)` const
- `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `ex & operator[] (const ex &index)`
- `ex & operator[] (size_t i)`
- `ex lhs ()` const  
*Left hand side of relational expression.*
- `ex rhs ()` const  
*Right hand side of relational expression.*
- `ex conjugate ()` const
- `ex real_part ()` const
- `ex imag_part ()` const
- `bool has (const ex &pattern, unsigned options=0)` const
- `bool find (const ex &pattern, exset &found)` const  
*Find all occurrences of a pattern.*
- `bool match (const ex &pattern)` const  
*Check whether expression matches a specified pattern.*

- `bool match` (const `ex` &pattern, `exmap` &repls) const
- `ex subs` (const `exmap` &m, unsigned options=0) const
- `ex subs` (const `lst` &ls, const `lst` &lr, unsigned options=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex subs` (const `ex` &e, unsigned options=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex map` (`map_function` &f) const
- `ex map` (`ex`(\*f)(const `ex` &e)) const
- void `accept` (`visitor` &v) const
- void `traverse_preorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, preorder traversal.*
- void `traverse_postorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, postorder traversal.*
- void `traverse` (`visitor` &v) const
- `bool is_polynomial` (const `ex` &vars) const
 

*Check whether expression is a polynomial.*
- `int degree` (const `ex` &s) const
- `int ldegree` (const `ex` &s) const
- `ex coeff` (const `ex` &s, int n=1) const
- `ex lcoeff` (const `ex` &s) const
- `ex tcoeff` (const `ex` &s) const
- `ex expand` (unsigned options=0) const
 

*Expand an expression.*
- `ex collect` (const `ex` &s, bool distributed=false) const
- `ex diff` (const `symbol` &s, unsigned nth=1) const
 

*Compute partial derivative of an expression.*
- `ex series` (const `ex` &r, int order, unsigned options=0) const
 

*Compute the truncated series expansion of an expression.*
- `ex normal` () const
 

*Normalization of rational functions.*
- `ex to_rational` (`exmap` &repl) const
 

*Rationalization of non-rational functions.*
- `ex to_polynomial` (`exmap` &repl) const
- `ex numer` () const
 

*Get numerator of an expression.*
- `ex denom` () const
 

*Get denominator of an expression.*
- `ex numer_denom` () const
 

*Get numerator and denominator of an expression.*
- `ex unit` (const `ex` &x) const
 

*Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $Q[x]$ .*
- `ex content` (const `ex` &x) const
 

*Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $Q[x]$ .*
- `numeric_integer_content` () const
 

*Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.*
- `ex primpart` (const `ex` &x) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex primpart` (const `ex` &x, const `ex` &cont) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$  when the content part is already known.*
- void `unitcontprim` (const `ex` &x, `ex` &u, `ex` &c, `ex` &p) const
 

*Compute unit part, content part, and primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex smod` (const `numeric` &xi) const

- [numeric max\\_coefficient](#) () const  
*Return maximum (absolute value) coefficient of a polynomial.*
- [exvector get\\_free\\_indices](#) () const
- [ex simplify\\_indexed](#) (unsigned options=0) const  
*Simplify/canonicalize expression containing indexed objects.*
- [ex simplify\\_indexed](#) (const [scalar\\_products](#) &sp, unsigned options=0) const  
*Simplify/canonicalize expression containing indexed objects.*
- int [compare](#) (const [ex](#) &other) const
- bool [is\\_equal](#) (const [ex](#) &other) const
- bool [is\\_zero](#) () const
- bool [is\\_zero\\_matrix](#) () const  
*Check whether expression is zero or zero matrix.*
- [ex symmetrize](#) () const  
*Symmetrize expression over its free indices.*
- [ex symmetrize](#) (const [lst](#) &l) const  
*Symmetrize expression over a list of objects (symbols, indices).*
- [ex antisymmetrize](#) () const  
*Antisymmetrize expression over its free indices.*
- [ex antisymmetrize](#) (const [lst](#) &l) const  
*Antisymmetrize expression over a list of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) () const  
*Symmetrize expression by cyclic permutation over its free indices.*
- [ex symmetrize\\_cyclic](#) (const [lst](#) &l) const  
*Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).*
- unsigned [return\\_type](#) () const
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- unsigned [gethash](#) () const

### Private Member Functions

- void [makewritable](#) ()  
*Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.*
- void [share](#) (const [ex](#) &other) const  
*Share equal objects between expressions.*

### Static Private Member Functions

- static [ptr](#)< [basic](#) > [construct\\_from\\_basic](#) (const [basic](#) &other)  
*Helper function for the ex-from-basic constructor.*
- static [basic](#) & [construct\\_from\\_int](#) (int i)
- static [basic](#) & [construct\\_from\\_uint](#) (unsigned int i)
- static [basic](#) & [construct\\_from\\_long](#) (long i)
- static [basic](#) & [construct\\_from\\_ulong](#) (unsigned long i)
- static [basic](#) & [construct\\_from\\_longlong](#) (long long i)
- static [basic](#) & [construct\\_from\\_ulonglong](#) (unsigned long long i)
- static [basic](#) & [construct\\_from\\_double](#) (double d)
- static [ptr](#)< [basic](#) > [construct\\_from\\_string\\_and\\_lst](#) (const std::string &s, const [ex](#) &l)

## Private Attributes

- `ptr< basic > bp`

*pointer to basic object managed by this*

## Friends

- class `archive_node`
- bool `are_ex_trivially_equal` (const `ex` &e1, const `ex` &e2)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- template<class T>  
const T & `ex_to` (const `ex` &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<class T>  
bool `is_a` (const `ex` &obj)  
*Check if ex is a handle to a T, including base classes.*
- template<class T>  
bool `is_exactly_a` (const `ex` &obj)  
*Check if ex is a handle to a T, not including base classes.*

## 8.48.1 Detailed Description

Lightweight wrapper for `GiNaC`'s symbolic objects.

It holds a pointer to the other object in order to do garbage collection by the method of reference counting. I.e., it is a smart pointer. Also, the constructor `ex::ex(const basic & other)` calls the methods that do automatic evaluation. E.g., `x-x` turns automatically into 0.

## 8.48.2 Constructor & Destructor Documentation

### 8.48.2.1 `ex()` [1/10]

```
GiNaC::ex::ex () [inline], [noexcept]
```

References `GiNaC::_num0_bp`, `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

Referenced by `antisymmetrize()`, `antisymmetrize()`, `archive_node`, `are_ex_trivially_equal`, `coeff()`, `collect()`, `compare()`, `conjugate()`, `construct_from_basic()`, `construct_from_string_and_lst()`, `content()`, `degree()`, `denom()`, `diff()`, `eval()`, `eval_integ()`, `eval_ncmul()`, `evalf()`, `evalm()`, `ex()`, `ex_to`, `expand()`, `find()`, `has()`, `imag_part()`, `is_a`, `is_equal()`, `is_exactly_a`, `is_polynomial()`, `is_zero()`, `is_zero_matrix()`, `lcoeff()`, `ldegree()`, `let_op()`, `lhs()`, `map()`, `map()`, `match()`, `match()`, `normal()`, `numer()`, `numer_denom()`, `op()`, `operator[]()`, `operator[]()`, `operator[]()`, `operator[]()`, `primpart()`, `primpart()`, `real_part()`, `rhs()`, `series()`, `share()`, `simplify_indexed()`, `simplify_indexed()`, `smod()`, `subs()`, `subs()`, `subs()`, `swap()`, `symmetrize()`, `symmetrize()`, `symmetrize_cyclic()`, `symmetrize_cyclic()`, `tcoeff()`, `to_polynomial()`, `to_rational()`, `unit()`, and `unitcontprim()`.

### 8.48.2.2 `ex()` [2/10]

```
GiNaC::ex::ex (
    const basic & other) [inline]
```

References `bp`, `construct_from_basic()`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

**8.48.2.3 ex()** [3/10]

```
GiNaC::ex::ex (  
    int i) [inline]
```

References [bp](#), [construct\\_from\\_int\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.4 ex()** [4/10]

```
GiNaC::ex::ex (  
    unsigned int i) [inline]
```

References [bp](#), [construct\\_from\\_uint\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.5 ex()** [5/10]

```
GiNaC::ex::ex (  
    long i) [inline]
```

References [bp](#), [construct\\_from\\_long\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.6 ex()** [6/10]

```
GiNaC::ex::ex (  
    unsigned long i) [inline]
```

References [bp](#), [construct\\_from\\_ulong\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.7 ex()** [7/10]

```
GiNaC::ex::ex (  
    long long i) [inline]
```

References [bp](#), [construct\\_from\\_longlong\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.8 ex()** [8/10]

```
GiNaC::ex::ex (  
    unsigned long long i) [inline]
```

References [bp](#), [construct\\_from\\_ulonglong\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**8.48.2.9 ex()** [9/10]

```
GiNaC::ex::ex (  
    double const d) [inline]
```

References [bp](#), [construct\\_from\\_double\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 8.48.2.10 `ex()` [10/10]

```
GiNaC::ex::ex (
    const std::string & s,
    const ex & l) [inline]
```

Construct `ex` from string and a list of symbols.

The input grammar is similar to the [GiNaC](#) output format. All symbols and indices to be used in the expression must be specified in a list in the second argument. Undefined symbols and other parser errors will throw an exception.

References [bp](#), [construct\\_from\\_string\\_and\\_lst\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [ex\(\)](#), and [GINAC\\_ASSERT](#).

## 8.48.3 Member Function Documentation

### 8.48.3.1 `swap()`

```
void GiNaC::ex::swap (
    ex & other) [inline], [noexcept]
```

Efficiently swap the contents of two expressions.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), [ex\(\)](#), [GINAC\\_ASSERT](#), and [swap\(\)](#).

Referenced by [GiNaC::ncmul::derivative\(\)](#), [GiNaC::ex\\_swap::operator\(\)\(\)](#), [swap\(\)](#), [GiNaC::swap\(\)](#), and [std::swap\(\)](#).

### 8.48.3.2 `begin()`

```
const_iterator GiNaC::ex::begin () const [inline], [noexcept]
```

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::antisymmetrize\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

### 8.48.3.3 `end()`

```
const_iterator GiNaC::ex::end () const [inline], [noexcept]
```

References [nops\(\)](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::log\\_expand\(\)](#), and [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#).

### 8.48.3.4 `preorder_begin()`

```
const_preorder_iterator GiNaC::ex::preorder_begin () const [inline]
```

References [nops\(\)](#).



**8.48.3.5 preorder\_end()**

```
const_preorder_iterator GiNaC::ex::preorder_end () const [inline], [noexcept]
```

**8.48.3.6 postorder\_begin()**

```
const_postorder_iterator GiNaC::ex::postorder_begin () const [inline]
```

References [nops\(\)](#).

**8.48.3.7 postorder\_end()**

```
const_postorder_iterator GiNaC::ex::postorder_end () const [inline], [noexcept]
```

**8.48.3.8 eval()**

```
ex GiNaC::ex::eval () const [inline]
```

References [bp](#), [eval\(\)](#), and [ex\(\)](#).

Referenced by [GiNaC::eval\(\)](#), and [eval\(\)](#).

**8.48.3.9 evalf()**

```
ex GiNaC::ex::evalf () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::subsvalue\(\)](#).

**8.48.3.10 evalm()**

```
ex GiNaC::ex::evalm () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::add::evalm\(\)](#), [GiNaC::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), and [is\\_zero\\_matrix\(\)](#).

**8.48.3.11 eval\_ncmul()**

```
ex GiNaC::ex::eval_ncmul (
    const exvector & v) const [inline]
```

References [bp](#), and [ex\(\)](#).

**8.48.3.12 eval\_integ()**

```
ex GiNaC::ex::eval_integ () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::eval\\_integ\(\)](#), and [GiNaC::integral::eval\\_integ\(\)](#).

**8.48.3.13 print()**

```
void GiNaC::ex::print (
    const print_context & c,
    unsigned level = 0) const
```

Print expression to stream.

The formatting of the output is determined by the kind of [print\\_context](#) object that is passed. Possible formattings include ginsh-parsable output (the default), tree-like output for debugging, and C++ source.

See also

[print\\_context](#)

References [bp](#).

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), and [GiNaC::zeta1\\_print\\_latex\(\)](#).

**8.48.3.14 dbgprint()**

```
void GiNaC::ex::dbgprint () const
```

Little wrapper around print to be called within a debugger.

References [bp](#).

**8.48.3.15 dbgprinttree()**

```
void GiNaC::ex::dbgprinttree () const
```

Little wrapper around printtree to be called within a debugger.

References [bp](#).

## 8.48.3.16 info()

```
bool GiNaC::ex::info (
    unsigned inf) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::abs\\_info\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_info\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan2\\_info\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_info\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::canonicalize\\_clifford\(\)](#), [GiNaC::clifford\\_moebius\\_map\(\)](#), [GiNaC::clifford\\_prime\(\)](#), [GiNaC::clifford\\_star\\_bar\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [content\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::csgn\\_series\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_eval\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticK\\_eval\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_info\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::func\\_arg\\_info\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::get\\_dim\\_uint\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::heur\\_gcd\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::iterated\\_integral2\\_eval\(\)](#), [GiNaC::iterated\\_integral3\\_eval\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::lgamma\\_eval\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_imag\\_part\(\)](#), [GiNaC::log\\_info\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::lst\\_to\\_clifford\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Order\\_imag\\_part\(\)](#), [GiNaC::Order\\_power\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::remove\\_dirac\\_ONE\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::step\\_series\(\)](#), [subs\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::tgamma\\_series\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::matrix::trace\(\)](#), [GiNaC::trig\\_info\(\)](#), [GiNaC::tryfactsubs\(\)](#), [unit\(\)](#), [unitcontprim\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zeta2\\_eval\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

## 8.48.3.17 nops()

```
size_t GiNaC::ex::nops () const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::clifford\\_star\\_bar\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::const\\_postorder\\_iterator\(\)](#), [GiNaC::dirac\\_trace\(\)](#), [GiNaC::divide\(\)](#), [end\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::idx\\_symmetrization\(\)](#), [GiNaC::Gt\\_detail::TransformExpressionWithC](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::nops\(\)](#), [postorder\\_begin\(\)](#), [preorder\\_begin\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::Gt::setArgs\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [GiNaC::indexed::simplify\\_indexed\\_product](#), [GiNaC::step\\_eval\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), and [GiNaC::zeta2\\_evalf\(\)](#).

### 8.48.3.18 op()

```
ex GiNaC::ex::op (
    size_t i) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::clifford\\_moebius\\_map\(\)](#), [GiNaC::clifford\\_prime\(\)](#), [GiNaC::clifford\\_star\\_bar\(\)](#), [GiNaC::clifford\\_to\\_lst\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::cliffordunit::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::decomp\\_rational\(\)](#), [denom\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::dirac\\_trace\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::Gt\\_detail::TransformExpressionWithCache< Type >::impl\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::lst\\_to\\_clifford\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [GiNaC::op\(\)](#), [GiNaC::ex\\_base\\_is\\_less::operator\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::remove\\_dirac\\_ONE\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::Gt::setArgs\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [GiNaC::indexed::simplify\\_indexed\\_product\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::spmapkey::spmapkey\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::step\\_eval\(\)](#), [subs\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::tryfactsubs\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), and [GiNaC::zeta2\\_deriv\(\)](#).

### 8.48.3.19 operator[]() [1/4]

```
ex GiNaC::ex::operator[] (
    const ex & index) const [inline]
```

References [bp](#), and [ex\(\)](#).

### 8.48.3.20 operator[]() [2/4]

```
ex GiNaC::ex::operator[] (
    size_t i) const [inline]
```

References [bp](#), and [ex\(\)](#).

### 8.48.3.21 let\_op()

```
ex & GiNaC::ex::let_op (
    size_t i)
```

Return modifiable operand/member at position i.

References [bp](#), [ex\(\)](#), and [makewriteable\(\)](#).

**8.48.3.22 operator[]() [3/4]**

```
ex & GiNaC::ex::operator[] (
    const ex & index)
```

References [bp](#), [ex\(\)](#), and [makewriteable\(\)](#).

**8.48.3.23 operator[]() [4/4]**

```
ex & GiNaC::ex::operator[] (
    size_t i)
```

References [bp](#), [ex\(\)](#), and [makewriteable\(\)](#).

**8.48.3.24 lhs()**

```
ex GiNaC::ex::lhs () const
```

Left hand side of relational expression.

References [bp](#), [ex\(\)](#), and [is\\_a](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::lhs\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::pseries::pseries\(\)](#).

**8.48.3.25 rhs()**

```
ex GiNaC::ex::rhs () const
```

Right hand side of relational expression.

References [bp](#), [ex\(\)](#), and [is\\_a](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::rhs\(\)](#).

**8.48.3.26 conjugate()**

```
ex GiNaC::ex::conjugate () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::conjugate\\_eval\(\)](#), [GiNaC::cos\\_conjugate\(\)](#), [GiNaC::cosh\\_conjugate\(\)](#), [GiNaC::exp\\_conjugate\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::sin\\_conjugate\(\)](#), [GiNaC::sinh\\_conjugate\(\)](#), [GiNaC::tan\\_conjugate\(\)](#), [GiNaC::tanh\\_conjugate\(\)](#), and [GiNaC::tgamma\\_conjugate\(\)](#).

**8.48.3.27 real\_part()**

```
ex GiNaC::ex::real_part () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::conjugate\\_real\\_part\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::real\\_part\(\)](#), and [GiNaC::real\\_part\\_eval\(\)](#).

**8.48.3.28 imag\_part()**

```
ex GiNaC::ex::imag_part () const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::conjugate\\_imag\\_part\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::imag\\_part\(\)](#), [GiNaC::imag\\_part\\_eval\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), and [GiNaC::log\\_conjugate\(\)](#).

**8.48.3.29 has()**

```
bool GiNaC::ex::has (
    const ex & pattern,
    unsigned options = 0) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::integral::expand\(\)](#), and [GiNaC::has\(\)](#).

**8.48.3.30 find()**

```
bool GiNaC::ex::find (
    const ex & pattern,
    exset & found) const
```

Find all occurrences of a pattern.

The found matches are appended to the "found" list. If the expression itself matches the pattern, the children are not further examined. This function returns true when any matches were found.

References [ex\(\)](#), [find\(\)](#), [match\(\)](#), [nops\(\)](#), and [op\(\)](#).

Referenced by [find\(\)](#), [GiNaC::find\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::idx::subs\(\)](#).

**8.48.3.31 match() [1/2]**

```
bool GiNaC::ex::match (
    const ex & pattern) const
```

Check whether expression matches a specified pattern.

References [bp](#), and [ex\(\)](#).

Referenced by [find\(\)](#), [GiNaC::match\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

**8.48.3.32 match()** [2/2]

```
bool GiNaC::ex::match (
    const ex & pattern,
    exmap & repls) const [inline]
```

References [bp](#), and [ex\(\)](#).

**8.48.3.33 subs()** [1/3]

```
ex GiNaC::ex::subs (
    const exmap & m,
    unsigned options = 0) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::canonicalize\\_clifford\(\)](#), [GiNaC::clifford\\_to\\_lst\(\)](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::csgn\\_series\(\)](#), [denom\(\)](#), [GiNaC::dirac\\_trace\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::basic::normal\(\)](#), [normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::indexed::reposition\\_dummy\\_indices\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::step\\_series\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::subsvalue\(\)](#), [GiNaC::symm\(\)](#), [GiNaC::symmetrize\\_cyclic\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), [GiNaC::tgamma\\_series\(\)](#), and [GiNaC::Gt::to\\_numeric\(\)](#).

**8.48.3.34 subs()** [2/3]

```
ex GiNaC::ex::subs (
    const lst & ls,
    const lst & lr,
    unsigned options = 0) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

References [GiNaC::container< C >::begin\(\)](#), [bp](#), [GiNaC::container< C >::end\(\)](#), [ex\(\)](#), [GINAC\\_ASSERT](#), [is\\_exactly\\_a](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), and [GiNaC::subs\\_options::pattern\\_is\\_p](#).

**8.48.3.35 subs()** [3/3]

```
ex GiNaC::ex::subs (
    const ex & e,
    unsigned options = 0) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

There are two valid types of replacement arguments: 1) a relational like `object==ex` and 2) a list of relationals `lst{object1==ex1,object2==ex2,...}`.

References [bp](#), [ex\(\)](#), [ex\\_to](#), [GINAC\\_ASSERT](#), [info\(\)](#), [is\\_a](#), [is\\_exactly\\_a](#), [GiNaC::info\\_flags::list](#), [op\(\)](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_product](#), and [GiNaC::info\\_flags::relation\\_equal](#).

**8.48.3.36 map() [1/2]**

```
ex GiNaC::ex::map (
    map_function & f) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::canonicalize\\_clifford\(\)](#), [GiNaC::clifford\\_prime\(\)](#), [GiNaC::clifford\\_star\\_bar\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::dirac\\_trace\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), and [GiNaC::remove\\_dirac\\_ONE\(\)](#).

**8.48.3.37 map() [2/2]**

```
ex GiNaC::ex::map (
    ex(* f) (const ex &e)) const
```

References [ex\(\)](#).

**8.48.3.38 accept()**

```
void GiNaC::ex::accept (
    visitor & v) const [inline]
```

References [bp](#).

Referenced by [traverse\\_postorder\(\)](#), and [traverse\\_preorder\(\)](#).

**8.48.3.39 traverse\_preorder()**

```
void GiNaC::ex::traverse_preorder (
    visitor & v) const
```

Traverse expression tree with given visitor, preorder traversal.

References [accept\(\)](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_preorder\(\)](#).

Referenced by [traverse\(\)](#), and [traverse\\_preorder\(\)](#).

**8.48.3.40 traverse\_postorder()**

```
void GiNaC::ex::traverse_postorder (
    visitor & v) const
```

Traverse expression tree with given visitor, postorder traversal.

References [accept\(\)](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_postorder\(\)](#).

Referenced by [traverse\\_postorder\(\)](#).



**8.48.3.41 traverse()**

```
void GiNaC::ex::traverse (
    visitor & v) const [inline]
```

References [traverse\\_preorder\(\)](#).

**8.48.3.42 is\_polynomial()**

```
bool GiNaC::ex::is_polynomial (
    const ex & vars) const
```

Check whether expression is a polynomial.

References [bp](#), [ex\(\)](#), [ex\\_to](#), and [is\\_a](#).

Referenced by [GiNaC::is\\_polynomial\(\)](#).

**8.48.3.43 degree()**

```
int GiNaC::ex::degree (
    const ex & s) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#), [content\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::integral::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [lcoeff\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::sprem\(\)](#), and [GiNaC::sr\\_gcd\(\)](#).

**8.48.3.44 ldegree()**

```
int GiNaC::ex::ldegree (
    const ex & s) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [content\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::ldegree\(\)](#), [GiNaC::mul::ldegree\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [tcoeff\(\)](#).

**8.48.3.45 coeff()**

```
ex GiNaC::ex::coeff (
    const ex & s,
    int n = 1) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [content\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [lcoeff\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), and [tcoeff\(\)](#).

### 8.48.3.46 lcoeff()

```
ex GiNaC::ex::lcoeff (
    const ex & s) const [inline]
```

References [coeff\(\)](#), [degree\(\)](#), and [ex\(\)](#).

Referenced by [content\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::sr\\_gcd\(\)](#), and [unit\(\)](#).

### 8.48.3.47 tcoeff()

```
ex GiNaC::ex::tcoeff (
    const ex & s) const [inline]
```

References [coeff\(\)](#), [ex\(\)](#), and [ldegree\(\)](#).

### 8.48.3.48 expand()

```
ex GiNaC::ex::expand (
    unsigned options = 0) const
```

Expand an expression.

#### Parameters

<i>options</i>	see <a href="#">GiNaC::expand_options</a>
----------------	---

References [bp](#), [ex\(\)](#), and [GiNaC::status\\_flags::expanded](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::clifford\\_star\\_bar\(\)](#), [GiNaC::color\\_trace\(\)](#), [content\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expand\\_map\\_function::operator\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::matrix::trace\(\)](#), [unit\(\)](#), and [unitcontprim\(\)](#).

### 8.48.3.49 collect()

```
ex GiNaC::ex::collect (
    const ex & s,
    bool distributed = false) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::matrix::charpoly\(\)](#), [GiNaC::basic::collect\(\)](#), and [GiNaC::collect\(\)](#).

### 8.48.3.50 diff()

```
ex GiNaC::ex::diff (
    const symbol & s,
    unsigned nth = 1) const
```

Compute partial derivative of an expression.

#### Parameters

<i>s</i>	symbol by which the expression is derived
----------	---

<i>nth</i>	order of derivative (default 1)
------------	---------------------------------

**Returns**

partial derivative as a new expression

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::diff\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_expl\\_derivative\(\)](#), [GiNaC::real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

**8.48.3.51 series()**

```
ex GiNaC::ex::series (
    const ex & r,
    int order,
    unsigned options = 0) const
```

Compute the truncated series expansion of an expression.

This function returns an expression containing an object of class pseries to represent the series. If the series does not terminate within the given truncation order, the last term of the series will be an order term.

**Parameters**

<i>r</i>	expansion relation, lhs holds variable and rhs holds point
<i>order</i>	truncation order of series calculations
<i>options</i>	of class <a href="#">series_options</a>

**Returns**

an expression holding a pseries object

References [GiNaC::\\_ex0](#), [bp](#), [ex\(\)](#), [ex\\_to](#), and [is\\_a](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::generalised\\_Bernoulli\\_nu](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_series\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

### 8.48.3.52 `normal()`

```
ex GiNaC::ex::normal () const
```

Normalization of rational functions.

This function converts an expression to its normal form "numerator/denominator", where numerator and denominator are (relatively prime) polynomials. Any subexpressions which are not rational functions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) are replaced by temporary symbols which are re-substituted by the (normalized) subexpressions before `normal()` returns (this way, any expression can be treated as a rational function). `normal()` is applied recursively to arguments of functions etc.

#### Returns

normalized expression

References `bp`, `ex()`, `GINAC_ASSERT`, `is_a`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::nops()`, `GiNaC::container< C >::op()`, `op()`, and `subs()`.

Referenced by `GiNaC::matrix::determinant()`, `GiNaC::add::normal()`, `GiNaC::mul::normal()`, `GiNaC::normal()`, `GiNaC::matrix::trace()`, and `GiNaC::Gt::zisToFundamental()`.

### 8.48.3.53 `to_rational()`

```
ex GiNaC::ex::to_rational (
    exmap & repl) const
```

Rationalization of non-rational functions.

This function converts a general expression to a rational function by replacing all non-rational subexpressions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) to temporary symbols. This makes it possible to use functions like `gcd()` and `divide()` on non-rational functions by applying `to_rational()` on the arguments, calling the desired function and re-substituting the temporary symbols in the result. To make the last step possible, all temporary symbols and their associated expressions are collected in the map specified by the `repl` parameter, ready to be passed as an argument to `ex::subs()`.

#### Parameters

<i>repl</i>	collects all temporary symbols and their replacements
-------------	---

#### Returns

rationalized expression

References `bp`, and `ex()`.

Referenced by `GiNaC::canonicalize_clifford()`, and `GiNaC::to_rational()`.

#### 8.48.3.54 to\_polynomial()

```
ex GiNaC::ex::to_polynomial (
    exmap & repl) const
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), and [GiNaC::to\\_polynomial\(\)](#).

#### 8.48.3.55 numer()

```
ex GiNaC::ex::numer () const
```

Get numerator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the numerator is returned.

See also

[ex::normal](#)

Returns

numerator

References [bp](#), [ex\(\)](#), [GINAC\\_ASSERT](#), [is\\_a](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::numer\(\)](#).

#### 8.48.3.56 denom()

```
ex GiNaC::ex::denom () const
```

Get denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the denominator is returned.

See also

[ex::normal](#)

Returns

denominator

References [bp](#), [ex\(\)](#), [GINAC\\_ASSERT](#), [is\\_a](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::denom\(\)](#).

**8.48.3.57 numer\_denom()**

```
ex GiNaC::ex::numer_denom () const
```

Get numerator and denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then a list [numerator, denominator] is returned.

See also

[ex::normal](#)

Returns

a list [numerator, denominator]

References [bp](#), [ex\(\)](#), [GINAC\\_ASSERT](#), [is\\_a](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::numer\\_denom\(\)](#).

**8.48.3.58 unit()**

```
ex GiNaC::ex::unit (
    const ex & x) const
```

Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

$x$	main variable
-----	---------------

Returns

unit part

See also

[ex::content](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [ex\(\)](#), [expand\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [info\(\)](#), [is\\_exactly\\_a](#), [lcoeff\(\)](#), [GiNaC::info\\_flags::negative](#), and [unit\(\)](#).

Referenced by [GiNaC::frac\\_cancel\(\)](#), [primpart\(\)](#), [unit\(\)](#), and [unitcontprim\(\)](#).

**8.48.3.59 content()**

```
ex GiNaC::ex::content (
    const ex & x) const
```

Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

$x$	main variable
-----	---------------

**Returns**

content part

**See also**

[ex::unit](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex0](#), [coeff\(\)](#), [degree\(\)](#), [ex\(\)](#), [expand\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [integer\\_content\(\)](#), [is\\_exactly\\_a](#), [is\\_zero\(\)](#), [lcoeff\(\)](#), [ldegree\(\)](#), and [GiNaC::info\\_flags::negative](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#), and [unitcontprim\(\)](#).

**8.48.3.60 integer\_content()**

```
numeric GiNaC::ex::integer_content () const
```

Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.

For a polynomial with rational coefficients, this returns  $g/l$  where  $g$  is the GCD of the coefficients' numerators and  $l$  is the LCM of the coefficients' denominators.

**Returns**

integer content

References [bp](#).

Referenced by [content\(\)](#), [GiNaC::gcd\(\)](#), and [GiNaC::heur\\_gcd\\_z\(\)](#).

**8.48.3.61 primpart()** [1/2]

```
ex GiNaC::ex::primpart (
    const ex & x) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The result will be a unit-normal polynomial with a content part of 1. The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

$x$	main variable
-----	---------------

**Returns**

primitive part

**See also**

[ex::unit](#), [ex::content](#), [ex::unitcontprim](#)

References [ex\(\)](#), and [unitcontprim\(\)](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#).

**8.48.3.62 primpart()** [2/2]

```
ex GiNaC::ex::primpart (
    const ex & x,
    const ex & c) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$  when the content part is already known.

This function is faster in computing the primitive part than the previous function.

**Parameters**

$x$	main variable
$c$	previously computed content part

**Returns**

primitive part

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [ex\(\)](#), [is\\_exactly\\_a](#), [is\\_zero\(\)](#), [GiNaC::quo\(\)](#), and [unit\(\)](#).

**8.48.3.63 unitcontprim()**

```
void GiNaC::ex::unitcontprim (
    const ex & x,
    ex & u,
    ex & c,
    ex & p) const
```

Compute unit part, content part, and primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of the three parts is the polynomial itself.

**Parameters**

$x$	main variable
-----	---------------



<i>u</i>	unit part (returned)
<i>c</i>	content part (returned)
<i>p</i>	primitive part (returned)

See also

[ex::unit](#), [ex::content](#), [ex::primpart](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::abs\(\)](#), [content\(\)](#), [ex\(\)](#), [ex\\_to](#), [expand\(\)](#), [info\(\)](#), [is\\_exactly\\_a](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::quo\(\)](#), and [unit\(\)](#).

Referenced by [GiNaC::gcd\(\)](#), and [primpart\(\)](#).

#### 8.48.3.64 smod()

```
ex GiNaC::ex::smod (
    const numeric & xi) const [inline]
```

References [bp](#), and [ex\(\)](#).

Referenced by [GiNaC::interpolate\(\)](#).

#### 8.48.3.65 max\_coefficient()

```
numeric GiNaC::ex::max_coefficient () const
```

Return maximum (absolute value) coefficient of a polynomial.

This function is used internally by [heur\\_gcd\(\)](#).

Returns

maximum coefficient

See also

[heur\\_gcd](#)

References [bp](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

#### 8.48.3.66 get\_free\_indices()

```
exvector GiNaC::ex::get_free_indices () const [inline]
```

References [bp](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::clifford\\_unit\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**8.48.3.67 simplify\_indexed()** [1/2]

```
ex GiNaC::ex::simplify_indexed (
    unsigned options = 0) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible and checks whether the free indices in sums are consistent.

**Parameters**

<i>options</i>	Simplification options (currently unused)
----------------	---

**Returns**

simplified expression

References [ex\(\)](#), and [GiNaC::simplify\\_indexed\(\)](#).

Referenced by [GiNaC::canonicalize\\_clifford\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), and [GiNaC::simplify\\_indexed\(\)](#).

**8.48.3.68 simplify\_indexed()** [2/2]

```
ex GiNaC::ex::simplify_indexed (
    const scalar_products & sp,
    unsigned options = 0) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible, checks whether the free indices in sums are consistent, and automatically replaces scalar products by known values if desired.

**Parameters**

<i>sp</i>	Scalar products to be replaced automatically
<i>options</i>	Simplification options (currently unused)

**Returns**

simplified expression

References [ex\(\)](#), and [GiNaC::simplify\\_indexed\(\)](#).

**8.48.3.69 compare()**

```
int GiNaC::ex::compare (
    const ex & other) const [inline]
```

References [bp](#), [ex\(\)](#), and [share\(\)](#).

Referenced by [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)\(\)](#), [GiNaC::ex\\_is\\_less::operator\(\)\(\)](#), [GiNaC::expair\\_rest\\_is\\_less::operator\(\)\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)\(\)](#), [GiNaC::terminfo\\_is\\_less::operator\(\)\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::operator\(\)\(\)](#), and [GiNaC::spmapkey::spmapkey\(\)](#).

**8.48.3.70 is\_equal()**

```
bool GiNaC::ex::is_equal (
    const ex & other) const [inline]
```

References [bp](#), [ex\(\)](#), and [share\(\)](#).

Referenced by [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::diracgamma::contract\\_with\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [is\\_zero\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::minimal\\_dim\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex\\_is\\_equal::operator\(\)](#), [GiNaC::idx\\_is\\_equal\\_ignore\\_dim::operator\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

**8.48.3.71 is\_zero()**

```
bool GiNaC::ex::is_zero () const [inline]
```

References [GiNaC::\\_ex0](#), [ex\(\)](#), and [is\\_equal\(\)](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::clifford\\_inverse\(\)](#), [GiNaC::clifford\\_to\\_lst\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [content\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::get\\_clifford\\_comp\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::is\\_zero\(\)](#), [is\\_zero\\_matrix\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li3\\_eval\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::operator<](#), [GiNaC::Gt\\_detail::pathintegral\\_term::operator==](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::Gt\\_detail::pathintegral\\_term::pathintegral\\_term\(\)](#), [GiNaC::prem\(\)](#), [primpart\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#), [GiNaC::indexed::simplify\\_indexed\\_product\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_yun\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), [unitcontprim\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

**8.48.3.72 is\_zero\_matrix()**

```
bool GiNaC::ex::is_zero_matrix () const
```

Check whether expression is zero or zero matrix.

References [evalm\(\)](#), [ex\(\)](#), [ex\\_to](#), [is\\_a](#), and [is\\_zero\(\)](#).

**8.48.3.73 symmetrize()** [1/2]

```
ex GiNaC::ex::symmetrize () const
```

Symmetrize expression over its free indices.

References [ex\(\)](#), [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\(\)](#).

Referenced by [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\(\)](#).

**8.48.3.74 symmetrize()** [2/2]

```
ex GiNaC::ex::symmetrize (
    const lst & l) const
```

Symmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\(\)](#), and [GiNaC::symm\(\)](#).

**8.48.3.75 antisymmetrize()** [1/2]

```
ex GiNaC::ex::antisymmetrize () const
```

Antisymmetrize expression over its free indices.

References [GiNaC::antisymmetrize\(\)](#), [ex\(\)](#), and [get\\_free\\_indices\(\)](#).

Referenced by [GiNaC::antisymmetrize\(\)](#), and [GiNaC::antisymmetrize\(\)](#).

**8.48.3.76 antisymmetrize()** [2/2]

```
ex GiNaC::ex::antisymmetrize (
    const lst & l) const
```

Antisymmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\(\)](#), and [GiNaC::symm\(\)](#).

**8.48.3.77 symmetrize\_cyclic()** [1/2]

```
ex GiNaC::ex::symmetrize_cyclic () const
```

Symmetrize expression by cyclic permutation over its free indices.

References [ex\(\)](#), [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::symmetrize\\_cyclic\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**8.48.3.78 symmetrize\_cyclic()** [2/2]

```
ex GiNaC::ex::symmetrize_cyclic (
    const lst & l) const
```

Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ex\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**8.48.3.79 return\_type()**

```
unsigned GiNaC::ex::return_type () const [inline]
```

References [bp](#).

Referenced by [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::exmul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), and [GiNaC::indexed::return\\_type\(\)](#).

**8.48.3.80 return\_type\_tinfo()**

```
return_type_t GiNaC::ex::return_type_tinfo () const [inline]
```

References [bp](#).

Referenced by [GiNaC::color\\_trace\(\)](#), [GiNaC::dirac\\_trace\(\)](#), and [GiNaC::indexed::return\\_type\\_tinfo\(\)](#).

**8.48.3.81 gethash()**

```
unsigned GiNaC::ex::gethash () const [inline]
```

References [bp](#).

Referenced by [GiNaC::function::calchash\(\)](#).

**8.48.3.82 construct\_from\_basic()**

```
ptr< basic > GiNaC::ex::construct_from_basic (
    const basic & other) [static], [private]
```

Helper function for the ex-from-basic constructor.

This is where [GiNaC](#)'s automatic evaluator and memory management are implemented.

See also

[ex::ex\(const basic &\)](#)

References [bp](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::basic::eval\(\)](#), [GiNaC::status\\_flags::evaluated](#), [ex\(\)](#), [GiNaC::basic::flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GINAC\\_ASSERT](#).

Referenced by [ex\(\)](#).

**8.48.3.83 construct\_from\_int()**

```
basic & GiNaC::ex::construct_from_int (
    int i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), [GiNaC::\\_num\\_9\\_p](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [construct\\_from\\_longlong\(\)](#), and [ex\(\)](#).

**8.48.3.84 construct\_from\_uint()**

```
basic & GiNaC::ex::construct_from_uint (
    unsigned int i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [construct\\_from\\_ulonglong\(\)](#), and [ex\(\)](#).

**8.48.3.85 construct\_from\_long()**

```
basic & GiNaC::ex::construct_from_long (
    long i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), [GiNaC::\\_num\\_9\\_p](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [ex\(\)](#).

**8.48.3.86 construct\_from\_ulong()**

```
basic & GiNaC::ex::construct_from_ulong (
    unsigned long i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [ex\(\)](#).

### 8.48.3.87 construct\_from\_longlong()

```
basic & GiNaC::ex::construct_from_longlong (
    long long i) [static], [private]
```

References [construct\\_from\\_int\(\)](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [ex\(\)](#).

### 8.48.3.88 construct\_from\_ulonglong()

```
basic & GiNaC::ex::construct_from_ulonglong (
    unsigned long long i) [static], [private]
```

References [construct\\_from\\_uint\(\)](#), and [GiNaC::dynallocate\(\)](#).

Referenced by [ex\(\)](#).

### 8.48.3.89 construct\_from\_double()

```
basic & GiNaC::ex::construct_from_double (
    double d) [static], [private]
```

References [GiNaC::dynallocate\(\)](#).

Referenced by [ex\(\)](#).

### 8.48.3.90 construct\_from\_string\_and\_lst()

```
ptr< basic > GiNaC::ex::construct_from_string_and_lst (
    const std::string & s,
    const ex & l) [static], [private]
```

References [ex\(\)](#).

Referenced by [ex\(\)](#).

### 8.48.3.91 makewriteable()

```
void GiNaC::ex::makewriteable () [private]
```

Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

Referenced by [let\\_op\(\)](#), [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

### 8.48.3.92 `share()`

```
void GiNaC::ex::share (
    const ex & other) const [private]
```

Share equal objects between expressions.

#### See also

`ex::compare(const ex &)`

References [bp](#), [ex\(\)](#), and [GiNaC::status\\_flags::not\\_shareable](#).

Referenced by [compare\(\)](#), and [is\\_equal\(\)](#).

## 8.48.4 Friends And Related Symbol Documentation

### 8.48.4.1 `archive_node`

```
friend class archive_node [friend]
```

References [archive\\_node](#), and [ex\(\)](#).

Referenced by [archive\\_node](#).

### 8.48.4.2 `are_ex_trivially_equal`

```
bool are_ex_trivially_equal (
    const ex & e1,
    const ex & e2) [friend]
```

Compare two objects of class quickly without doing a deep tree traversal.

#### Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

References [bp](#), and [ex\(\)](#).



### 8.48.4.3 ex\_to

```
template<class T>
const T & ex_to (
    const ex & e) [friend]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of e first. Also, you shouldn't cache the returned reference because [GiNaC's](#) garbage collector may destroy the referenced object any time it's used in another expression.

#### Parameters

<i>e</i>	expression
----------	------------

#### Returns

reference to object of class T

#### See also

[is\\_exactly\\_a<class T>\(\)](#)

References [bp](#), [ex\(\)](#), [GINAC\\_ASSERT](#), and [is\\_a](#).

Referenced by [is\\_polynomial\(\)](#), [is\\_zero\\_matrix\(\)](#), [series\(\)](#), [subs\(\)](#), and [unitcontprim\(\)](#).

### 8.48.4.4 is\_a

```
template<class T>
bool is_a (
    const ex & obj) [friend]
```

Check if ex is a handle to a T, including base classes.

References [bp](#), [ex\(\)](#), and [GiNaC::is\\_a\(\)](#).

Referenced by [denom\(\)](#), [ex\\_to](#), [is\\_polynomial\(\)](#), [is\\_zero\\_matrix\(\)](#), [lhs\(\)](#), [normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [rhs\(\)](#), [series\(\)](#), and [subs\(\)](#).

### 8.48.4.5 is\_exactly\_a

```
template<class T>
bool is_exactly_a (
    const ex & obj) [friend]
```

Check if ex is a handle to a T, not including base classes.

References [bp](#), [ex\(\)](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

Referenced by [content\(\)](#), [primpart\(\)](#), [subs\(\)](#), [subs\(\)](#), [unit\(\)](#), and [unitcontprim\(\)](#).

## 8.48.5 Member Data Documentation

### 8.48.5.1 bp

```
ptr<basic> GiNaC::ex::bp [mutable], [private]
```

pointer to basic object managed by this

Referenced by [accept\(\)](#), [GiNaC::archive\\_node::archive\\_node\(\)](#), [are\\_ex\\_trivially\\_equal](#), [coeff\(\)](#), [collect\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [construct\\_from\\_basic\(\)](#), [dbgprint\(\)](#), [dbgprinttree\(\)](#), [degree\(\)](#), [denom\(\)](#), [diff\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\\_to](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [gethash\(\)](#), [has\(\)](#), [GiNaC::archive\\_node::has\\_same\\_ex\\_as\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [is\\_a](#), [is\\_equal\(\)](#), [is\\_exactly\\_a](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [lhs\(\)](#), [makewriteable\(\)](#), [map\(\)](#), [match\(\)](#), [match\(\)](#), [max\\_coefficient\(\)](#), [nops\(\)](#), [normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [print\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), [series\(\)](#), [share\(\)](#), [smod\(\)](#), [subs\(\)](#), [subs\(\)](#), [subs\(\)](#), [swap\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [ex.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symmetry.cpp](#)

## 8.49 GiNaC::ex\_base\_is\_less Struct Reference

### Public Member Functions

- [bool operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### 8.49.1 Member Function Documentation

#### 8.49.1.1 operator()()

```
bool GiNaC::ex_base_is_less::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::is\\_a\(\)](#), and [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 8.50 GiNaC::ex\_is\_equal Struct Reference

```
#include <ex.h>
```

## Public Member Functions

- `bool operator() (const ex &lh, const ex &rh) const`

## 8.50.1 Member Function Documentation

### 8.50.1.1 operator()()

```
bool GiNaC::ex_is_equal::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 8.51 GiNaC::ex\_is\_less Struct Reference

```
#include <ex.h>
```

## Public Member Functions

- `bool operator() (const ex &lh, const ex &rh) const`

## 8.51.1 Member Function Documentation

### 8.51.1.1 operator()()

```
bool GiNaC::ex_is_less::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::ex::compare\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 8.52 GiNaC::ex\_swap Struct Reference

```
#include <ex.h>
```

## Public Member Functions

- void [operator\(\)](#) ([ex](#) &lh, [ex](#) &rh) const

## 8.52.1 Member Function Documentation

### 8.52.1.1 [operator\(\)](#)

```
void GiNaC::ex_swap::operator() (
    ex & lh,
    ex & rh) const [inline]
```

References [GiNaC::ex::swap\(\)](#).

The documentation for this struct was generated from the following file:

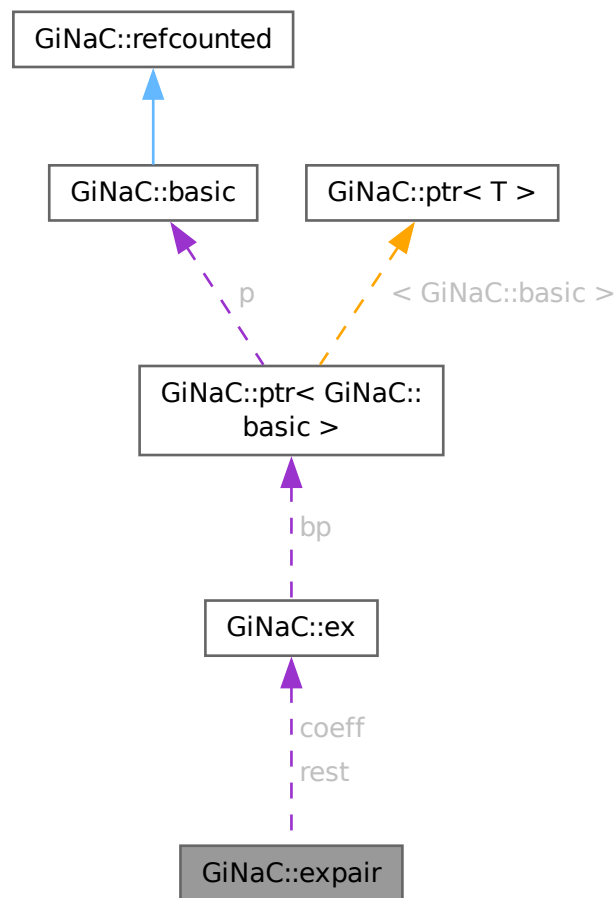
- [ex.h](#)

## 8.53 GiNaC::expair Class Reference

A pair of expressions.

```
#include <expair.h>
```

Collaboration diagram for GiNaC::expair:



## Public Member Functions

- `expair ()`
- `expair (const ex &r, const ex &c)`  
Construct an `expair` from two `ex`.
- `bool is_equal (const expair &other) const`  
Member-wise check for canonical ordering equality.
- `bool is_less (const expair &other) const`  
Member-wise check for canonical ordering lessness.
- `int compare (const expair &other) const`  
Member-wise check for canonical ordering.
- `void print (std::ostream &os) const`
- `bool is_canonical_numeric () const`  
True if this is of the form `(numeric,ex(1))`.
- `void swap (expair &other)`  
Swap contents with other `expair`.
- `const expair conjugate () const`

## Public Attributes

- [ex rest](#)  
*first member of pair, an arbitrary expression*
- [ex coeff](#)  
*second member of pair, must be numeric*

### 8.53.1 Detailed Description

A pair of expressions.

This is similar to STL's `pair<>`. It is slightly extended since we need to account for methods like `.compare()`. Also, since this is meant for use by class `expairseq` it must satisfy the invariance that the member `coeff` must be of type `numeric`.

### 8.53.2 Constructor & Destructor Documentation

#### 8.53.2.1 `expair()` [1/2]

```
GiNaC::expair::expair () [inline]
```

References [coeff](#), and [rest](#).

Referenced by [compare\(\)](#), [conjugate\(\)](#), [is\\_equal\(\)](#), [is\\_less\(\)](#), and [swap\(\)](#).

#### 8.53.2.2 `expair()` [2/2]

```
GiNaC::expair::expair (
    const ex & r,
    const ex & c) [inline]
```

Construct an `expair` from two `ex`.

References [coeff](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [rest](#).

### 8.53.3 Member Function Documentation

#### 8.53.3.1 `is_equal()`

```
bool GiNaC::expair::is_equal (
    const expair & other) const [inline]
```

Member-wise check for canonical ordering equality.

References [coeff](#), [expair\(\)](#), and [rest](#).

Referenced by [GiNaC::conjugateepvector\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), and [GiNaC::expairseq::subschildren\(\)](#).

### 8.53.3.2 is\_less()

```
bool GiNaC::expair::is_less (
    const expair & other) const [inline]
```

Member-wise check for canonical ordering lessness.

References [coeff](#), [expair\(\)](#), and [rest](#).

Referenced by [GiNaC::expair\\_is\\_less::operator\(\)](#).

### 8.53.3.3 compare()

```
int GiNaC::expair::compare (
    const expair & other) const [inline]
```

Member-wise check for canonical ordering.

References [coeff](#), [expair\(\)](#), and [rest](#).

### 8.53.3.4 print()

```
void GiNaC::expair::print (
    std::ostream & os) const
```

References [coeff](#), [GiNaC::print\\_tree::delta\\_indent](#), and [rest](#).

### 8.53.3.5 is\_canonical\_numeric()

```
bool GiNaC::expair::is_canonical_numeric () const [inline]
```

True if this is of the form (numeric,ex(1)).

References [coeff](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [rest](#).

### 8.53.3.6 swap()

```
void GiNaC::expair::swap (
    expair & other) [inline]
```

Swap contents with other expair.

References [coeff](#), [expair\(\)](#), and [rest](#).

Referenced by [GiNaC::mul::derivative\(\)](#), [GiNaC::expair\\_swap::operator\(\)](#), and [GiNaC::swap\(\)](#).

### 8.53.3.7 conjugate()

```
const expair GiNaC::expair::conjugate () const
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [coeff](#), [expair\(\)](#), and [rest](#).

Referenced by [GiNaC::conjugateepvector\(\)](#).

## 8.53.4 Member Data Documentation

### 8.53.4.1 rest

```
ex GiNaC::expair::rest
```

first member of pair, an arbitrary expression

Referenced by [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [expair\(\)](#), [expair\(\)](#), [is\\_canonical\\_numeric\(\)](#), [is\\_equal\(\)](#), [is\\_less\(\)](#), [GiNaC::expair\\_rest\\_is\\_less::operator\(\)\(\)](#), [print\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), and [swap\(\)](#).

### 8.53.4.2 coeff

```
ex GiNaC::expair::coeff
```

second member of pair, must be numeric

Referenced by [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [expair\(\)](#), [expair\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [is\\_canonical\\_numeric\(\)](#), [is\\_equal\(\)](#), [is\\_less\(\)](#), [print\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following files:

- [expair.h](#)
- [expair.cpp](#)

## 8.54 GiNaC::expair\_is\_less Struct Reference

Function object for insertion into third argument of STL's sort() etc.

```
#include <expair.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [expair](#) &lh, const [expair](#) &rh) const



### 8.54.1 Detailed Description

Function object for insertion into third argument of STL's sort() etc.

### 8.54.2 Member Function Documentation

#### 8.54.2.1 operator()

```
bool GiNaC::expair_is_less::operator() (
    const expair & lh,
    const expair & rh) const [inline]
```

References [GiNaC::expair::is\\_less\(\)](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 8.55 GiNaC::expair\_rest\_is\_less Struct Reference

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

```
#include <expair.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [expair](#) &lh, const [expair](#) &rh) const

### 8.55.1 Detailed Description

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

Note that this does not define a strict weak ordering since for any symbol  $x$  we have neither  $3*x < 2*x$  or  $2*x < 3*x$ . Handle with care!

### 8.55.2 Member Function Documentation

#### 8.55.2.1 operator()

```
bool GiNaC::expair_rest_is_less::operator() (
    const expair & lh,
    const expair & rh) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::expair::rest](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 8.56 GiNaC::expair\_swap Struct Reference

```
#include <expair.h>
```

### Public Member Functions

- void [operator\(\)](#) ([expair](#) &lh, [expair](#) &rh) const

### 8.56.1 Member Function Documentation

#### 8.56.1.1 operator()

```
void GiNaC::expair_swap::operator() (
    expair & lh,
    expair & rh) const [inline]
```

References [GiNaC::expair::swap\(\)](#).

The documentation for this struct was generated from the following file:

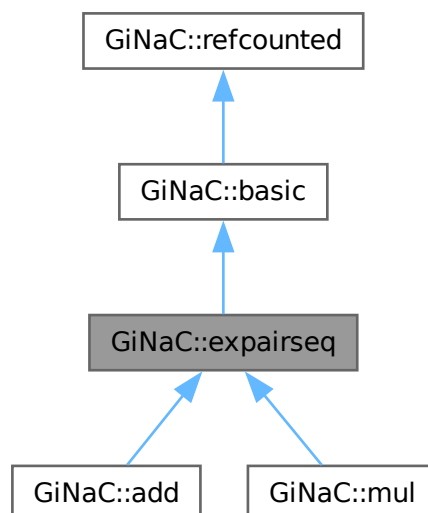
- [expair.h](#)

## 8.57 GiNaC::expairseq Class Reference

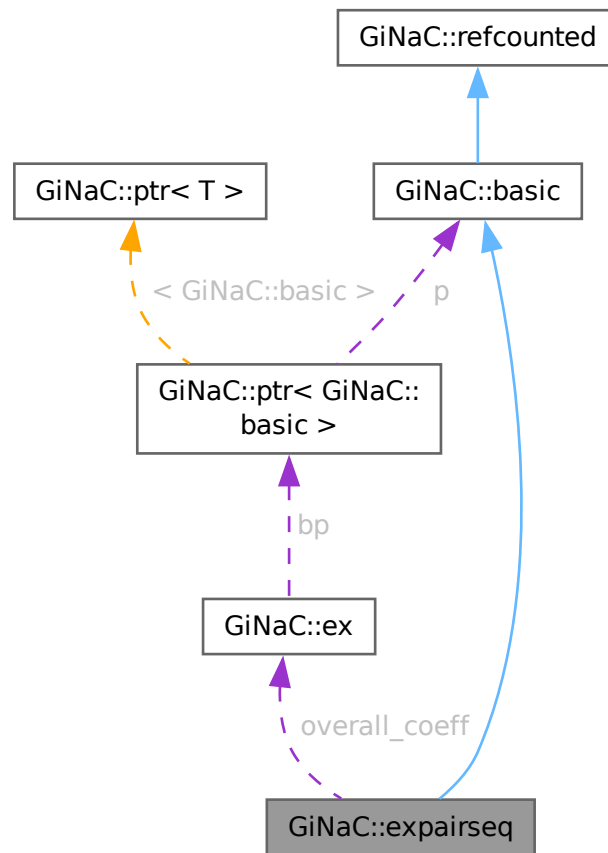
A sequence of class [expair](#).

```
#include <expairseq.h>
```

Inheritance diagram for [GiNaC::expairseq](#):



Collaboration diagram for GiNaC::expairseq:



## Public Member Functions

- `expairseq` (const `ex` &lh, const `ex` &rh)
- `expairseq` (const `exvector` &v)
- `expairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `expairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- bool `info` (unsigned inf) const override  
Information about the object.
- size\_t `nops` () const override  
Number of operands/members.
- `ex op` (size\_t i) const override  
Return operand/member at position i.
- `ex map` (`map_function` &f) const override  
Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).
- `ex eval` () const override  
Perform coefficient-wise automatic term rewriting rules in this class.

- `ex to_rational (exmap &repl)` const override  
*Implementation of `ex::to_rational()` for `expairseqs`.*
- `ex to_polynomial (exmap &repl)` const override  
*Implementation of `ex::to_polynomial()` for `expairseqs`.*
- `bool match (const ex &pattern, exmap &repl_lst)` const override  
*Check whether the expression matches a given pattern.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex conjugate ()` const override
- `void archive (archive_node &n)` const override  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate ()` const  
*Create a clone of this object on the heap.*
- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual `void dbgprint ()` const  
*Little wrapper around `print` to be called within a debugger.*
- virtual `void dbgprinttree ()` const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position `i`.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object `s`.*

- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `return_type` () const override
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- virtual `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false) const  
*Create an object of this type.*
- virtual `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const
- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- virtual `expair split_ex_to_pair` (const `ex` &e) const  
*Form an expair from an ex, using the corresponding semantics.*
- virtual `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const
- virtual `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const
- virtual `ex recombine_pair_to_ex` (const `expair` &p) const  
*Form an ex out of an expair, using the corresponding semantics.*
- virtual bool `expair_needs_further_processing` (`epp` it)
- virtual `ex default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expair` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expairseq` (const `expairseq` &s1, const `expairseq` &s2)
- void `construct_from_expairseq_ex` (const `expairseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
*Combine this expairseq with argument exvector.*
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
*Combine this expairseq with argument epvector.*
- void `canonicalize` ()  
*Brings this expairseq into a sorted (canonical) form.*
- void `combine_same_terms_sorted_seq` ()  
*Compact a presorted expairseq by combining all matching expairs to one each.*
- bool `is_canonical` () const  
*Check if this expairseq is in sorted (canonical) form.*
- `epvector expandchildren` (unsigned options) const  
*Member-wise expand the expairs in this sequence.*
- `epvector evalchildren` () const  
*Member-wise evaluate the expairs in this sequence.*
- `epvector subschildren` (const `exmap` &m, unsigned options=0) const  
*Member-wise substitute in this sequence.*

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [epvector seq](#)
- [ex overall\\_coeff](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.57.1 Detailed Description

A sequence of class `expair`.

This is used for time-critical classes like sums and products of terms since handling a list of coeff and rest is much faster than handling a list of products or powers, respectively. (Not incidentally, Maple does it the same way, maybe others too.) The semantics is (at least) twofold: one for addition and one for multiplication and several methods have to be overridden by derived classes to reflect the change in semantics. However, most functionality turns out to be shared between addition and multiplication, which is the reason why there is this base class.

### 8.57.2 Constructor & Destructor Documentation

#### 8.57.2.1 `expairseq()` [1/4]

```
GiNaC::expairseq::expairseq (
    const ex & lh,
    const ex & rh)
```

References [construct\\_from\\_2\\_ex\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

Referenced by [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [make\\_flat\(\)](#), [make\\_flat\(\)](#), [thisexpairseq\(\)](#), and [thisexpairseq\(\)](#).

### 8.57.2.2 `expairseq()` [2/4]

```
GiNaC::expairseq::expairseq (
    const exvector & v)
```

References [construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 8.57.2.3 `expairseq()` [3/4]

```
GiNaC::expairseq::expairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false)
```

References [construct\\_from\\_epvector\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [is\\_canonical\(\)](#), and [overall\\_coeff](#).

### 8.57.2.4 `expairseq()` [4/4]

```
GiNaC::expairseq::expairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false)
```

References [construct\\_from\\_epvector\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [is\\_canonical\(\)](#), and [overall\\_coeff](#).

## 8.57.3 Member Function Documentation

### 8.57.3.1 `precedence()`

```
unsigned GiNaC::expairseq::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

Referenced by [do\\_print\(\)](#), and [printpair\(\)](#).

### 8.57.3.2 `info()`

```
bool GiNaC::expairseq::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::flags](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).



### 8.57.3.3 nops()

```
size_t GiNaC::expairseq::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::mul::has\(\)](#), [match\(\)](#), and [GiNaC::Order\\_eval\(\)](#).

### 8.57.3.4 op()

```
ex GiNaC::expairseq::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [match\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::add::series\(\)](#), and [GiNaC::mul::series\(\)](#).

### 8.57.3.5 map()

```
ex GiNaC::expairseq::map (
    map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_a\(\)](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 8.57.3.6 eval()

```
ex GiNaC::expairseq::eval () const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::dynallocate\(\)](#), [evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::basic::flags](#), [GiNaC::basic::hold\(\)](#), and [overall\\_coeff](#).

### 8.57.3.7 to\_rational()

```
ex GiNaC::expairseq::to_rational (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [to\\_rational\(\)](#).

Referenced by [to\\_rational\(\)](#).

### 8.57.3.8 to\_polynomial()

```
ex GiNaC::expairseq::to_polynomial (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [to\\_polynomial\(\)](#).

Referenced by [to\\_polynomial\(\)](#).

### 8.57.3.9 match()

```
bool GiNaC::expairseq::match (
    const ex & pattern,
    exmap & repl_lst) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [nops\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 8.57.3.10 subs()

```
ex GiNaC::expairseq::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The `ex` returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::no\\_index](#), [overall\\_coeff](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [subschildren\(\)](#), and [thisexpairseq\(\)](#).

### 8.57.3.11 conjugate()

```
ex GiNaC::expairseq::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::basic::ex](#), [overall\\_coeff](#), [seq](#), and [thisexpairseq\(\)](#).

### 8.57.3.12 archive()

```
void GiNaC::expairseq::archive (
    archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [overall\\_coeff](#), and [seq](#).

### 8.57.3.13 read\_archive()

```
void GiNaC::expairseq::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [canonicalize\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_b](#), [GiNaC::archive\\_node::find\\_property\\_range\(\)](#), [GINAC\\_ASSERT](#), [is\\_canonical\(\)](#), [overall\\_coeff](#), and [seq](#).

### 8.57.3.14 is\_equal\_same\_type()

```
bool GiNaC::expairseq::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [expairseq\(\)](#), [overall\\_coeff](#), and [seq](#).

### 8.57.3.15 return\_type()

```
unsigned GiNaC::expairseq::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 8.57.3.16 calchash()

```
unsigned GiNaC::expairseq::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [overall\\_coeff](#), [GiNaC::rotate\\_left\(\)](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).

### 8.57.3.17 expand()

```
ex GiNaC::expairseq::expand (
    unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::ex](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [overall\\_coeff](#), [GiNaC::basic::setflag\(\)](#), and [thisexpairseq\(\)](#).

**8.57.3.18 thisexpairseq() [1/2]**

```
ex GiNaC::expairseq::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false) const [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::basic::ex](#), and [expairseq\(\)](#).

Referenced by [conjugate\(\)](#), [expand\(\)](#), [map\(\)](#), [match\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**8.57.3.19 thisexpairseq() [2/2]**

```
ex GiNaC::expairseq::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::basic::ex](#), and [expairseq\(\)](#).

**8.57.3.20 printseq()**

```
void GiNaC::expairseq::printseq (
    const print\_context & c,
    char delim,
    unsigned this_precedence,
    unsigned upper_precedence) const [protected], [virtual]
```

References [default\\_overall\\_coeff\(\)](#), [overall\\_coeff](#), [printpair\(\)](#), [GiNaC::print\\_context::s](#), and [seq](#).

Referenced by [do\\_print\(\)](#).

**8.57.3.21 printpair()**

```
void GiNaC::expairseq::printpair (
    const print\_context & c,
    const expair & p,
    unsigned upper_precedence) const [protected], [virtual]
```

References [GiNaC::expair::coeff](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::expair::rest](#), and [GiNaC::print\\_context::s](#).

Referenced by [is\\_canonical\(\)](#), and [printseq\(\)](#).

**8.57.3.22 split\_ex\_to\_pair()**

```
expair GiNaC::expairseq::split_ex_to_pair (
    const ex & e) const [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::\\_ex1](#), and [GiNaC::basic::ex](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [match\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**8.57.3.23 combine\_ex\_with\_coeff\_to\_pair()**

```
expair GiNaC::expairseq::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

Referenced by [evalchildren\(\)](#), and [subschildren\(\)](#).

**8.57.3.24 combine\_pair\_with\_coeff\_to\_pair()**

```
expair GiNaC::expairseq::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::expair::coeff](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::expair::rest](#).

**8.57.3.25 recombine\_pair\_to\_ex()**

```
ex GiNaC::expairseq::recombine_pair_to_ex (
    const expair & p) const [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::expair::coeff](#), [GiNaC::basic::ex](#), and [GiNaC::expair::rest](#).

Referenced by [map\(\)](#), [op\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**8.57.3.26 expair\_needs\_further\_processing()**

```
bool GiNaC::expairseq::expair_needs_further_processing (
    expair it) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex1](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

Referenced by [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), and [construct\\_from\\_expairseq\\_ex\(\)](#).

**8.57.3.27 default\_overall\_coeff()**

```
ex GiNaC::expairseq::default_overall_coeff () const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex0](#), and [GiNaC::basic::ex](#).

Referenced by [do\\_print\\_tree\(\)](#), [map\(\)](#), [match\(\)](#), [nops\(\)](#), [op\(\)](#), [printseq\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**8.57.3.28 combine\_overall\_coeff() [1/2]**

```
void GiNaC::expairseq::combine_overall_coeff (
    const ex & c) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [overall\\_coeff](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [make\\_flat\(\)](#), and [make\\_flat\(\)](#).

**8.57.3.29 combine\_overall\_coeff() [2/2]**

```
void GiNaC::expairseq::combine_overall_coeff (
    const ex & c1,
    const ex & c2) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [overall\\_coeff](#).

**8.57.3.30 can\_make\_flat()**

```
bool GiNaC::expairseq::can_make_flat (
    const expair & p) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

**8.57.3.31 do\_print()**

```
void GiNaC::expairseq::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [precedence\(\)](#), [printseq\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.57.3.32 do\_print\_tree()**

```
void GiNaC::expairseq::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [default\\_overall\\_coeff\(\)](#), [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [nops\(\)](#), [overall\\_coeff](#), [GiNaC::print\\_context::s](#), and [seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.57.3.33 construct\_from\_2\_ex()**

```
void GiNaC::expairseq::construct_from_2_ex (
    const ex & lh,
    const ex & rh) [protected]
```

References [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::ex::compare\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), and [GiNaC::mul::mul\(\)](#).

**8.57.3.34 construct\_from\_2\_expairseq()**

```
void GiNaC::expairseq::construct_from_2_expairseq (
    const expairseq & s1,
    const expairseq & s2) [protected]
```

References [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [GiNaC::ex\\_to\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expairseq\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).

**8.57.3.35 construct\_from\_expairseq\_ex()**

```
void GiNaC::expairseq::construct_from_expairseq_ex (
    const expairseq & s,
    const ex & e) [protected]
```

References [GiNaC::class\\_info< OPT >::first](#), [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expairseq\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [overall\\_coeff](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).



**8.57.3.36 construct\_from\_exvector()**

```
void GiNaC::expairseq::construct_from_exvector (
    const exvector & v) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), and [GiNaC::mul::mul\(\)](#).

**8.57.3.37 construct\_from\_epvector() [1/2]**

```
void GiNaC::expairseq::construct_from_epvector (
    const epvector & v,
    bool do_index_renaming = false) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), and [GiNaC::mul::mul\(\)](#).

**8.57.3.38 construct\_from\_epvector() [2/2]**

```
void GiNaC::expairseq::construct_from_epvector (
    epvector && v,
    bool do_index_renaming = false) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

**8.57.3.39 make\_flat() [1/2]**

```
void GiNaC::expairseq::make_flat (
    const exvector & v) [protected]
```

Combine this expairseq with argument exvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::\\_ex1](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expairseq\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [overall\\_coeff](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**8.57.3.40 make\_flat() [2/2]**

```
void GiNaC::expairseq::make_flat (
    const epvector & v,
    bool do_index_renaming = false) [protected]
```

Combine this expairseq with argument epvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::\\_ex1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expairseq\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::is\\_a\(\)](#), [overall\\_coeff](#), and [seq](#).

**8.57.3.41 canonicalize()**

```
void GiNaC::expairseq::canonicalize () [protected]
```

Brings this expairseq into a sorted (canonical) form.

References [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_exvector\(\)](#), and [read\\_archive\(\)](#).

**8.57.3.42 combine\_same\_terms\_sorted\_seq()**

```
void GiNaC::expairseq::combine_same_terms_sorted_seq () [protected]
```

Compact a presorted expairseq by combining all matching expairs to one each.

On an add object, this is responsible for  $2*x+3*x+y \rightarrow 5*x+y$ , for instance.

References [construct\\_from\\_epvector\(\)](#), [GiNaC::ex\\_to\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), and [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**8.57.3.43 is\_canonical()**

```
bool GiNaC::expairseq::is_canonical () const [protected]
```

Check if this expairseq is in sorted (canonical) form.

Useful mainly for debugging or in assertions since being sorted is an invariance.

References [GiNaC::is\\_exactly\\_a\(\)](#), [printpair\(\)](#), and [seq](#).

Referenced by [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), and [read\\_archive\(\)](#).

**8.57.3.44 expandchildren()**

```
epvector GiNaC::expairseq::expandchildren (
    unsigned options) const [protected]
```

Member-wise expand the expairs in this sequence.

See also

[expairseq::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::ex](#), and [seq](#).

Referenced by [GiNaC::add::expand\(\)](#), and [expand\(\)](#).

### 8.57.3.45 evalchildren()

```
epvector GiNaC::expairseq::evalchildren () const [protected]
```

Member-wise evaluate the expairs in this sequence.

See also

[expairseq::eval\(\)](#)

Returns

epvector containing evaluated pairs, empty if no members had to be changed.

References [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [seq](#), and [unlikely](#).

Referenced by [GiNaC::add::eval\(\)](#), [eval\(\)](#), and [GiNaC::mul::eval\(\)](#).

### 8.57.3.46 subschildren()

```
epvector GiNaC::expairseq::subschildren (
    const exmap & m,
    unsigned options = 0) const [protected]
```

Member-wise substitute in this sequence.

See also

[expairseq::subs\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::basic::ex](#), [GiNaC::expair::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_pr](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [subs\(\)](#).

## 8.57.4 Member Data Documentation

### 8.57.4.1 seq

```
epvector GiNaC::expairseq::seq [protected]
```

Referenced by [archive\(\)](#), [calchash\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [canonicalize\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::add::derivative\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::add::eval\\_ncmul\(\)](#), [GiNaC::mul::eval\\_ncmul\(\)](#), [evalchildren\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_canonical\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::add::is\\_polynomial\(\)](#), [GiNaC::mul::is\\_polynomial\(\)](#), [GiNaC::add::ldegree\(\)](#), [GiNaC::mul::ldegree\(\)](#), [make\\_flat\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [nops\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [op\(\)](#), [GiNaC::add::print\\_add\(\)](#), [printseq\(\)](#), [read\\_archive\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::add::return\\_type\(\)](#), [GiNaC::mul::return\\_type\(\)](#), [GiNaC::add::return\\_type\\_tinfo\(\)](#), [GiNaC::mul::return\\_type\\_tinfo\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 8.57.4.2 overall\_coeff

ex GiNaC::expairseq::overall\_coeff [protected]

Referenced by [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [combine\\_overall\\_coeff\(\)](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [GiNaC::add::expand\(\)](#), [expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::add::ldegree\(\)](#), [make\\_flat\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [nops\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [op\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [printseq\(\)](#), [read\\_archive\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

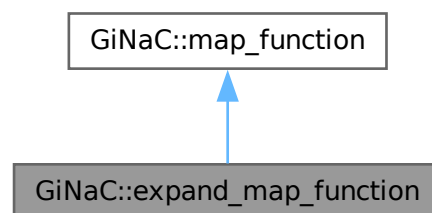
The documentation for this class was generated from the following files:

- [expairseq.h](#)
- [expairseq.cpp](#)
- [normal.cpp](#)

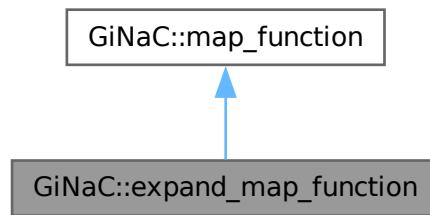
## 8.58 GiNaC::expand\_map\_function Struct Reference

Function object to be applied by [basic::expand\(\)](#).

Inheritance diagram for GiNaC::expand\_map\_function:



Collaboration diagram for GiNaC::expand\_map\_function:



### Public Member Functions

- [expand\\_map\\_function](#) (unsigned o)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Public Attributes

- unsigned [options](#)

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.58.1 Detailed Description

Function object to be applied by [basic::expand\(\)](#).

## 8.58.2 Constructor & Destructor Documentation

### 8.58.2.1 [expand\\_map\\_function\(\)](#)

```
GiNaC::expand_map_function::expand_map_function (
    unsigned o) [inline]
```

References [options](#).

### 8.58.3 Member Function Documentation

#### 8.58.3.1 operator()

```
ex GiNaC::expand_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::ex::expand\(\)](#), and [options](#).

### 8.58.4 Member Data Documentation

#### 8.58.4.1 options

```
unsigned GiNaC::expand_map_function::options
```

Referenced by [expand\\_map\\_function\(\)](#), and [operator\(\)](#).

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 8.59 GiNaC::expand\_options Class Reference

Flags to control the behavior of [expand\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum { [expand\\_indexed](#) = 0x0001 , [expand\\_function\\_args](#) = 0x0002 , [expand\\_rename\\_idx](#) = 0x0004 , [expand\\_transcendental](#) = 0x0008 }

#### 8.59.1 Detailed Description

Flags to control the behavior of [expand\(\)](#).

### 8.59.2 Member Enumeration Documentation

#### 8.59.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<a href="#">expand_indexed</a>	expands (a+b).i to a.i+b.i
--------------------------------	----------------------------

expand_function_args	expands the arguments of functions
expand_rename_idx	used internally by <a href="#">mul::expand()</a>
expand_transcendental	expands transcendental functions like log and exp

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.60 GiNaC::factor\_options Class Reference

Flags to control the polynomial factorization.

```
#include <flags.h>
```

### Public Types

- enum { [polynomial](#) = 0x0000 , [all](#) = 0x0001 }

### 8.60.1 Detailed Description

Flags to control the polynomial factorization.

### 8.60.2 Member Enumeration Documentation

#### 8.60.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

polynomial	factor only expressions that are polynomials
all	factor all polynomial subexpressions

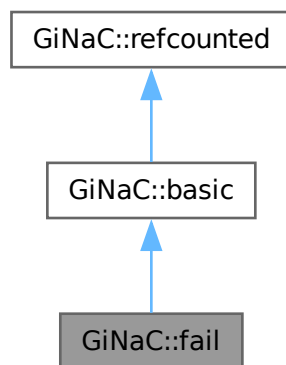
The documentation for this class was generated from the following file:

- [flags.h](#)

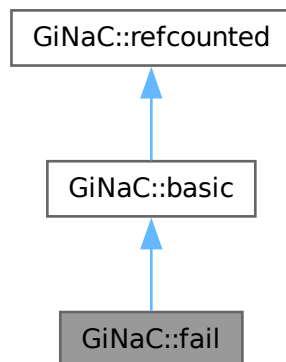
## 8.61 GiNaC::fail Class Reference

```
#include <fail.h>
```

Inheritance diagram for GiNaC::fail:



Collaboration diagram for GiNaC::fail:



### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const



## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*

- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*

- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.61.1 Member Function Documentation

### 8.61.1.1 `return_type()`

```
unsigned GiNaC::fail::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 8.61.1.2 `do_print()`

```
void GiNaC::fail::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

The documentation for this class was generated from the following file:

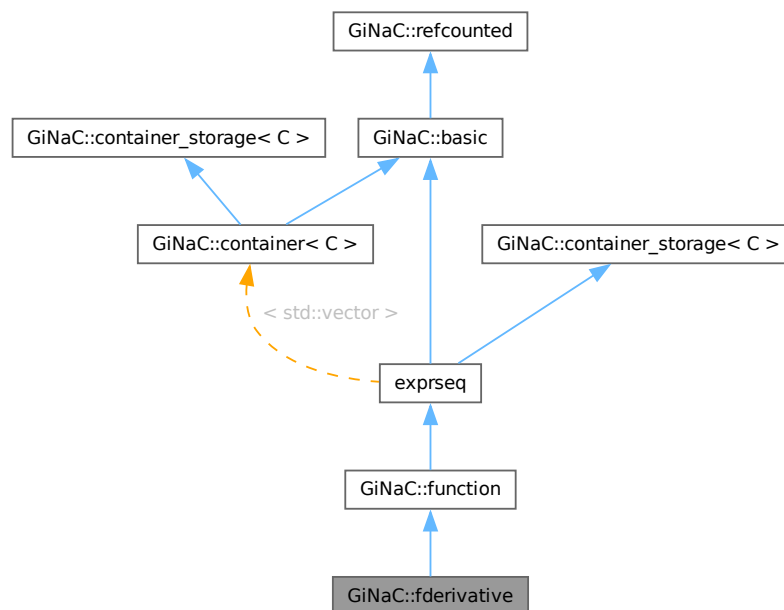
- [fail.h](#)

## 8.62 GiNaC::fderivative Class Reference

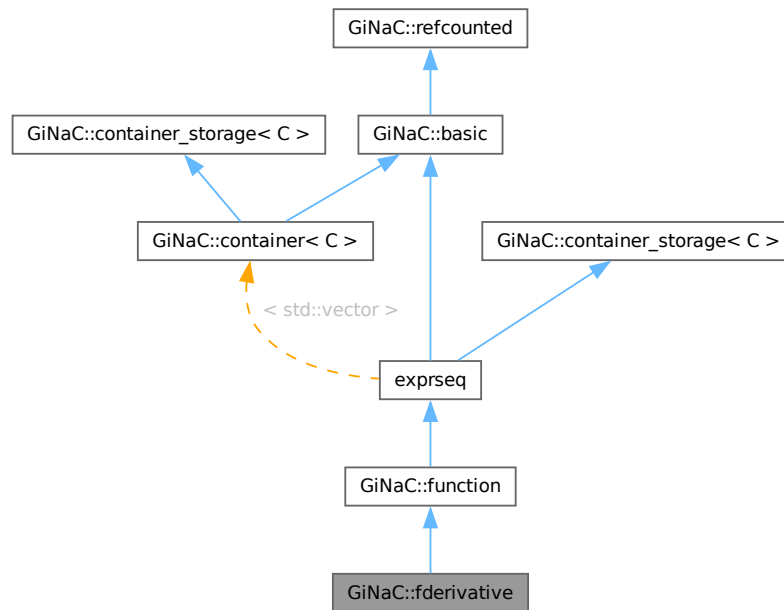
This class represents the (abstract) derivative of a symbolic function.

```
#include <fderivative.h>
```

Inheritance diagram for GiNaC::fderivative:



Collaboration diagram for GiNaC::fderivative:



## Public Member Functions

- **fderivative** (unsigned ser, unsigned param, const **exvector** &args)  
*Construct derivative with respect to one parameter.*
- **fderivative** (unsigned ser, const **paramset** &params, const **exvector** &args)  
*Construct derivative with respect to multiple parameters.*
- **fderivative** (unsigned ser, const **paramset** &params, **exvector** &&v)
- void **print** (const **print\_context** &c, unsigned level=0) const override  
*Output to stream.*
- **ex eval** () const override  
*Perform automatic non-interruptive term rewriting rules.*
- **ex series** (const **relational** &r, int order, unsigned options=0) const override  
*The series expansion of derivatives falls back to Taylor expansion.*
- **ex thiscontainer** (const **exvector** &v) const override  
*Similar to **duplicate()**, but with a preset sequence.*
- **ex thiscontainer** (**exvector** &&v) const override  
*Similar to **duplicate()**, but with a preset sequence (which gets pilfered).*
- void **archive** (**archive\_node** &n) const override  
*Archive the object.*
- void **read\_archive** (const **archive\_node** &n, **lst** &syms) override  
*Load (deserialize) the object from an archive node.*
- const **paramset** & **derivatives** () const  
*Expose this object's derivative structure.*

## Public Member Functions inherited from `GiNaC::function`

- `function` (unsigned ser)
- `function` (unsigned ser, const `ex` &param1)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12, const `ex` &param13)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12, const `ex` &param13, const `ex` &param14)
- `function` (unsigned ser, const `exprseq` &es)
- `function` (unsigned ser, const `exvector` &v)
- `function` (unsigned ser, `exvector` &&v)
- void `print` (const `print_context` &c, unsigned level=0) const override  
*Output to stream.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- `ex eval_ncmul` (const `exvector` &v) const override  
*This method is defined to be in line with behavior of `function::return_type()`.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Implementation of `ex::series` for functions.*
- `ex thiscontainer` (const `exvector` &v) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&v) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*

- `ex conjugate ()` const override  
*Implementation of `ex::conjugate` for functions.*
- `ex real_part ()` const override  
*Implementation of `ex::real_part` for functions.*
- `ex imag_part ()` const override  
*Implementation of `ex::imag_part` for functions.*
- `void archive (archive_node &n)` const override  
*Archive the object.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Construct object from `archive_node`.*
- `bool info (unsigned inf)` const override  
*Implementation of `ex::info` for functions.*
- `ex power (const ex &exp)` const
- `unsigned get_serial ()` const
- `std::string get_name ()` const  
*Return the print name of the function.*

## Public Member Functions inherited from `GiNaC::container< std::vector >`

- `container (STLT const &s)`
- `container (STLT &&v)`
- `container (exvector::const_iterator b, exvector::const_iterator e)`
- `container (std::initializer_list< ex > il)`
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position *i*.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position *i*.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend (const ex &b)`  
*Add element at front.*
- `container & append (const ex &b)`  
*Add element at back.*
- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin ()` const
- `const_iterator end ()` const
- `const_reverse_iterator rbegin ()` const
- `const_reverse_iterator rend ()` const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- const `basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `dbgprint () const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object `s`.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object `s`.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree `n` in object `s`.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`



- *Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- *Try to contract two indexed expressions that appear in the same product.*
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override
- *Implementation of `ex::diff()` for derivatives.*
- bool `is_equal_same_type` (const `basic` &other) const override
- *Returns true if two objects of same type are equal.*
- bool `match_same_type` (const `basic` &other) const override
- *Returns true if the attributes of two objects are similar enough for a match.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_context` &c, unsigned level) const
- void `do_print_csrx` (const `print_csrx` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::function](#)

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for functions.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex pderivative](#) (unsigned diff\_param) const
- [ex expl\\_derivative](#) (const [symbol](#) &s) const
- bool [lookup\\_remember\\_table](#) ([ex](#) &result) const
- void [store\\_remember\\_table](#) ([ex](#) const &result) const

## Protected Member Functions inherited from [GiNaC::container](#)< [std::vector](#) >

- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned options=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage](#)< [C](#) >

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

**Protected Attributes**

- [paramset parameter\\_set](#)

*Set of parameter numbers with respect to which to take the derivative.*

**Protected Attributes inherited from [GiNaC::function](#)**

- unsigned [serial](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**Additional Inherited Members****Public Types inherited from [GiNaC::container< std::vector >](#)**

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

**Static Public Member Functions inherited from [GiNaC::function](#)**

- static unsigned [register\\_new](#) ([function\\_options](#) const &opt)
- static unsigned [find\\_function](#) (const std::string &name, unsigned nparams)  
*Find serial number of function by name and number of parameters.*
- static std::vector< [function\\_options](#) > [get\\_registered\\_functions](#) ()

**Static Public Attributes inherited from [GiNaC::function](#)**

- static unsigned [current\\_serial](#) = 0  
*This can be used as a hook for external applications.*

**Protected Types inherited from [GiNaC::container< std::vector >](#)**

- typedef [container\\_storage< std::vector >::STLT](#) [STLT](#)

**Protected Types inherited from [GiNaC::container\\_storage< C >](#)**

- typedef C< [ex](#) > [STLT](#)

## Static Protected Member Functions inherited from [GiNaC::function](#)

- static `std::vector< function\_options > & registered_functions ()`

## Static Protected Member Functions inherited from [GiNaC::container< std::vector >](#)

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

### 8.62.1 Detailed Description

This class represents the (abstract) derivative of a symbolic function.

It is used to represent the derivatives of functions that do not have a derivative or series expansion procedure defined.

### 8.62.2 Constructor & Destructor Documentation

#### 8.62.2.1 `fderivative()` [1/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    unsigned param,
    const exvector & args)
```

Construct derivative with respect to one parameter.

#### Parameters

<i>ser</i>	Serial number of function
<i>param</i>	Number of parameter with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

References [GiNaC::function::function\(\)](#), and [parameter\\_set](#).

Referenced by [derivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 8.62.2.2 `fderivative()` [2/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    const exvector & args)
```

Construct derivative with respect to multiple parameters.

#### Parameters

<i>ser</i>	Serial number of function
------------	---------------------------

<i>params</i>	Set of numbers of parameters with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

References [GiNaC::function::function\(\)](#), and [parameter\\_set](#).

### 8.62.2.3 fderivative() [3/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    exvector && v)
```

References [GiNaC::function::function\(\)](#), and [parameter\\_set](#).

## 8.62.3 Member Function Documentation

### 8.62.3.1 print()

```
void GiNaC::fderivative::print (
    const print_context & c,
    unsigned level = 0) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

#### Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::print\(\)](#).

### 8.62.3.2 eval()

```
ex GiNaC::fderivative::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::function::function\(\)](#), [GiNaC::basic::hold\(\)](#), [parameter\\_set](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 8.62.3.3 series()

```
ex GiNaC::fderivative::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

The series expansion of derivatives falls back to Taylor expansion.

See also

[basic::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::series\(\)](#).

### 8.62.3.4 thiscontainer() [1/2]

```
ex GiNaC::fderivative::thiscontainer (
    const exvector & v) const [override], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

### 8.62.3.5 thiscontainer() [2/2]

```
ex GiNaC::fderivative::thiscontainer (
    exvector && v) const [override], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

### 8.62.3.6 archive()

```
void GiNaC::fderivative::archive (
    archive_node & n) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [GiNaC::container< std::vector >::end\(\)](#), and [parameter\\_set](#).

### 8.62.3.7 read\_archive()

```
void GiNaC::fderivative::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::archive\\_node::find\\_unsigned\(\)](#), and [parameter\\_set](#).

### 8.62.3.8 derivative()

```
ex GiNaC::fderivative::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for derivatives.

It applies the chain rule.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [fderivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [parameter\\_set](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 8.62.3.9 is\_equal\_same\_type()

```
bool GiNaC::fderivative::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< std::vector >](#).

References [fderivative\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [parameter\\_set](#).

### 8.62.3.10 match\_same\_type()

```
bool GiNaC::fderivative::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [fderivative\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [parameter\\_set](#).

### 8.62.3.11 derivatives()

```
const paramset & GiNaC::fderivative::derivatives () const
```

Expose this object's derivative structure.

Parameter numbers occurring more than once stand for repeated differentiation with respect to that parameter. If a symbolic function  $f(x,y)$  is differentiated with respect to  $x$ , this method will return  $\{0\}$ . If  $f(x,y)$  is differentiated twice with respect to  $y$ , it will return  $\{1,1\}$ . (This corresponds to the way this object is printed.)

Returns

multiset of function's parameter numbers that are abstractly differentiated.

References [parameter\\_set](#).

### 8.62.3.12 do\_print()

```
void GiNaC::fderivative::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [GiNaC::container< std::vector >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< std::vector >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< std::vector >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::function::serial](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).



### 8.62.3.13 do\_print\_latex()

```
void GiNaC::fderivative::do_print_latex (
    const print\_context & c,
    unsigned level) const [protected]
```

References [GiNaC::container< std::vector >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< std::vector >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< std::vector >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::function::serial](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.62.3.14 do\_print\_csrc()

```
void GiNaC::fderivative::do_print_csrc (
    const print\_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::container< std::vector >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< std::vector >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< std::vector >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::function::serial](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.62.3.15 do\_print\_tree()

```
void GiNaC::fderivative::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::container< std::vector >::end\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::container< std::vector >::nops\(\)](#), [parameter\\_set](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.62.4 Member Data Documentation

### 8.62.4.1 parameter\_set

```
paramset GiNaC::fderivative::parameter_set [protected]
```

Set of parameter numbers with respect to which to take the derivative.

Referenced by [archive\(\)](#), [derivative\(\)](#), [derivatives\(\)](#), [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [fderivative\(\)](#), [fderivative\(\)](#), [fderivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

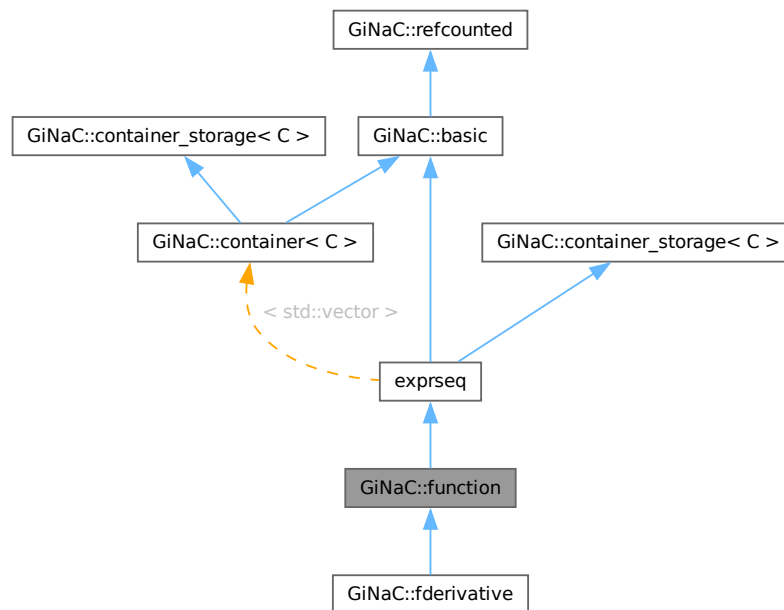
- [fderivative.h](#)
- [fderivative.cpp](#)

## 8.63 GiNaC::function Class Reference

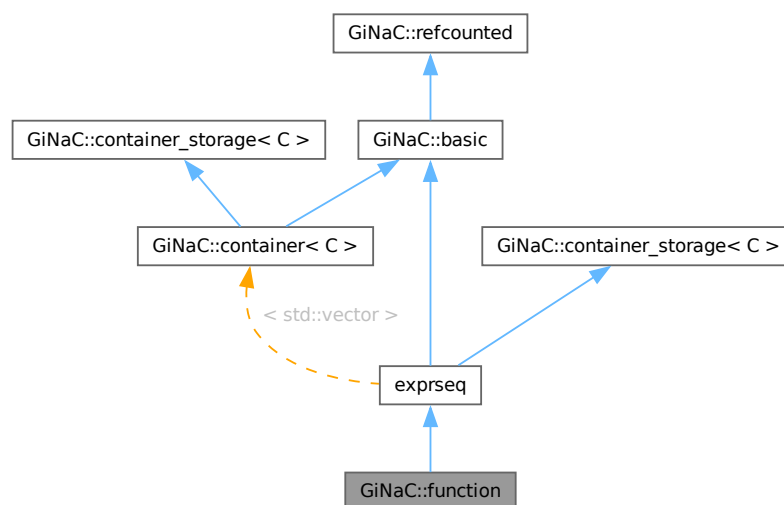
The class function is used to implement builtin functions like sin, cos... and user defined functions.

```
#include <function.h>
```

Inheritance diagram for GiNaC::function:



Collaboration diagram for GiNaC::function:



## Public Member Functions

- [function](#) (unsigned ser)
- [function](#) (unsigned ser, const [ex](#) &param1)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13, const [ex](#) &param14)
- [function](#) (unsigned ser, const [exprseq](#) &es)
- [function](#) (unsigned ser, const [exvector](#) &v)
- [function](#) (unsigned ser, [exvector](#) &&v)
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- [ex expand](#) (unsigned options=0) const override  
*Expand expression, i.e.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*This method is defined to be in line with behavior of [function::return\\_type\(\)](#).*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const override  
*Implementation of [ex::series](#) for functions.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- [ex thiscontainer](#) ([exvector](#) &&v) const override  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*

- [ex conjugate](#) () const override  
*Implementation of [ex::conjugate](#) for functions.*
- [ex real\\_part](#) () const override  
*Implementation of [ex::real\\_part](#) for functions.*
- [ex imag\\_part](#) () const override  
*Implementation of [ex::imag\\_part](#) for functions.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Archive the object.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Construct object from [archive\\_node](#).*
- bool [info](#) (unsigned inf) const override  
*Implementation of [ex::info](#) for functions.*
- [ex power](#) (const [ex](#) &exp) const
- unsigned [get\\_serial](#) () const
- std::string [get\\_name](#) () const  
*Return the print name of the function.*

## Public Member Functions inherited from [GiNaC::container](#)< [std::vector](#) >

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex](#) & [let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [container](#) & [prepend](#) (const [ex](#) &b)  
*Add element at front.*
- [container](#) & [append](#) (const [ex](#) &b)  
*Add element at back.*
- [container](#) & [remove\\_first](#) ()  
*Remove first element.*
- [container](#) & [remove\\_last](#) ()  
*Remove last element.*
- [container](#) & [remove\\_all](#) ()  
*Remove all elements.*
- [container](#) & [sort](#) ()  
*Sort elements.*
- [container](#) & [unique](#) ()  
*Remove adjacent duplicate elements.*
- [const\\_iterator begin](#) () const
- [const\\_iterator end](#) () const
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic(const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate() const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm() const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ() const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed(const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void dbgprint() const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree() const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has(const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match(const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map(map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept(GiNaC::visitor &v) const`
- virtual `bool is_polynomial(const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree(const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree(const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff(const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex collect(const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal(exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational(exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial(exmap &repl) const`
- virtual `numeric integer_content() const`
- virtual `ex smod(const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient() const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices() const`

- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

  - template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - `ex subs_one_level` (const `exmap` &m, unsigned options) const

*Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*

  - const `basic` & `hold` () const

*Stop further evaluation.*

  - unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*

  - const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Static Public Member Functions

- static unsigned `register_new` (`function_options` const &opt)
  - static unsigned `find_function` (const std::string &name, unsigned nparams)
- Find serial number of function by name and number of parameters.*
- static std::vector< `function_options` > `get_registered_functions` ()

## Static Public Attributes

- static unsigned `current_serial` = 0
- This can be used as a hook for external applications.*

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for functions.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex pderivative` (unsigned `diff_param`) const
- `ex expl_derivative` (const `symbol` &s) const
- bool `lookup_remember_table` (`ex` &result) const
- void `store_remember_table` (`ex` const &result) const

### Protected Member Functions inherited from `GiNaC::container< std::vector >`

- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned `this_`←  
`_precedence`, unsigned `upper_precedence`=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT` `subchildren` (const `exmap` &m, unsigned options=0) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from `GiNaC::container_storage< C >`

- `container_storage` ()
- `container_storage` (size\_t n, const `ex` &e)
- `container_storage` (std::initializer\_list< `ex` > il)
- template<class In>  
`container_storage` (In b, In e)
- void `reserve` (size\_t)
- `~container_storage` ()
- void `reserve` (size\_t n)
- void `reserve` (std::vector< `ex` > &v, size\_t n)

**Static Protected Member Functions**

- static std::vector< [function\\_options](#) > & [registered\\_functions](#) ()

**Static Protected Member Functions inherited from [GiNaC::container< std::vector >](#)**

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) (STLT &, size\_t)

**Protected Attributes**

- unsigned [serial](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**Friends**

- class [remember\\_table\\_entry](#)

**Additional Inherited Members****Public Types inherited from [GiNaC::container< std::vector >](#)**

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

**Protected Types inherited from [GiNaC::container< std::vector >](#)**

- typedef [container\\_storage< std::vector >::STLT](#) STLT



## Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > [STLT](#)

### 8.63.1 Detailed Description

The class function is used to implement builtin functions like sin, cos... and user defined functions.

### 8.63.2 Constructor & Destructor Documentation

#### 8.63.2.1 function() [1/18]

```
GiNaC::function::function (
    unsigned ser)
```

References [serial](#).

Referenced by [GiNaC::fderivative::eval\(\)](#), [evalf\(\)](#), [GiNaC::fderivative::fderivative\(\)](#), [GiNaC::fderivative::fderivative\(\)](#), [GiNaC::fderivative::fderivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [remember\\_table\\_entry](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 8.63.2.2 function() [2/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1)
```

References [serial](#).

#### 8.63.2.3 function() [3/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2)
```

References [serial](#).

#### 8.63.2.4 function() [4/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3)
```

References [serial](#).

#### 8.63.2.5 function() [5/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4)
```

References [serial](#).

#### 8.63.2.6 function() [6/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5)
```

References [serial](#).

#### 8.63.2.7 function() [7/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6)
```

References [serial](#).

#### 8.63.2.8 function() [8/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7)
```

References [serial](#).

### 8.63.2.9 function() [9/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8)
```

References [serial](#).

### 8.63.2.10 function() [10/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9)
```

References [serial](#).

### 8.63.2.11 function() [11/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10)
```

References [serial](#).

#### 8.63.2.12 function() [12/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11)
```

References [serial](#).

#### 8.63.2.13 function() [13/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12)
```

References [serial](#).

#### 8.63.2.14 function() [14/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13)
```

References [serial](#).

**8.63.2.15 function()** [15/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13,
    const ex & param14)
```

References [serial](#).

**8.63.2.16 function()** [16/18]

```
GiNaC::function::function (
    unsigned ser,
    const exprseq & es)
```

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), and [serial](#).

**8.63.2.17 function()** [17/18]

```
GiNaC::function::function (
    unsigned ser,
    const exvector & v)
```

References [serial](#).

**8.63.2.18 function()** [18/18]

```
GiNaC::function::function (
    unsigned ser,
    exvector && v)
```

References [serial](#).

**8.63.3 Member Function Documentation****8.63.3.1 print()**

```
void GiNaC::function::print (
    const print_context & c,
    unsigned level = 0) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

**Parameters**

<i>c</i>	print context object that describes the output formatting
----------	---

<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting
--------------	---

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::basic::flags](#), [GiNaC::print\\_context\\_options::get\\_id\(\)](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hashvalue](#), [GiNaC::is\\_a\(\)](#), [GiNaC::function\\_options::name](#), [GiNaC::container< std::vector >::nops\(\)](#), [GiNaC::function\\_options::nparams](#), [GiNaC::class\\_info< OPT >::options](#), [GiNaC::container< std::vector >::precedence\(\)](#), [precedence\(\)](#), [print\(\)](#), [GiNaC::function\\_options::print\\_dispatch\\_table](#), [GiNaC::function\\_options::print\\_use\\_exvector\\_args](#), [GiNaC::container< std::vector >::printseq\(\)](#), [registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::TeX\\_name](#).

Referenced by [print\(\)](#), and [remember\\_table\\_entry](#).

### 8.63.3.2 precedence()

```
unsigned GiNaC::function::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< std::vector >](#).

References [precedence\(\)](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [precedence\(\)](#), and [print\(\)](#).

### 8.63.3.3 expand()

```
ex GiNaC::function::expand (
    unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::expand\\_f](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::function\\_options::expand\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::expanded](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 8.63.3.4 eval()

```
ex GiNaC::function::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [current\\_serial](#), [GiNaC::function\\_options::eval\\_f](#), [GiNaC::function\\_options::eval\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lookup\\_remember\\_table\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [store\\_remember\\_table\(\)](#), [GiNaC::function\\_options::symtree](#), [thiscontainer\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

### 8.63.3.5 evalf()

```
ex GiNaC::function::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::evalf\\_f](#), [GiNaC::function\\_options::evalf\\_params\\_first](#), [GiNaC::function\\_options::evalf\\_use\\_exvector\\_args](#), [function\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [GiNaC::iterated\\_integral2\\_eval\(\)](#), and [GiNaC::iterated\\_integral3\\_eval\(\)](#).

### 8.63.3.6 eval\_ncmul()

```
ex GiNaC::function::eval_ncmul (
    const exvector & v) const [override], [virtual]
```

This method is defined to be in line with behavior of [function::return\\_type\(\)](#).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

### 8.63.3.7 calchash()

```
unsigned GiNaC::function::calchash () const [override], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::container< std::vector >::nops\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), [GiNaC::rotate\\_left\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 8.63.3.8 series()

```
ex GiNaC::function::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series](#) for functions.

@see [ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [GiNaC::basic::series\(\)](#), [GiNaC::function\\_options::series\\_f](#), and [GiNaC::function\\_options::series\\_use\\_exvector\\_args](#).

**8.63.3.9 thiscontainer() [1/2]**

```
ex GiNaC::function::thiscontainer (
    const exvector & v) const \[override\], \[virtual\]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [function\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**8.63.3.10 thiscontainer() [2/2]**

```
ex GiNaC::function::thiscontainer (
    exvector && v) const \[override\], \[virtual\]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [function\(\)](#), and [serial](#).

**8.63.3.11 conjugate()**

```
ex GiNaC::function::conjugate () const \[override\], \[virtual\]
```

Implementation of [ex::conjugate](#) for functions.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::function\\_options::conjugate\\_f](#), [GiNaC::function\\_options::conjugate\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**8.63.3.12 real\_part()**

```
ex GiNaC::function::real_part () const \[override\], \[virtual\]
```

Implementation of [ex::real\\_part](#) for functions.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::basic::real\\_part\(\)](#), [GiNaC::function\\_options::real\\_part\\_f](#), [GiNaC::function\\_options::real\\_part\\_use\\_exvector\\_args](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).



### 8.63.3.13 `imag_part()`

```
ex GiNaC::function::imag_part () const [override], [virtual]
```

Implementation of [ex::imag\\_part](#) for functions.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::imag\\_part\(\)](#), [GiNaC::function\\_options::imag\\_part\\_f](#), [GiNaC::function\\_options::imag\\_part](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

### 8.63.3.14 `archive()`

```
void GiNaC::function::archive (
    archive_node & n) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::archive\\_node::add\\_string\(\)](#), [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), and [serial](#).

### 8.63.3.15 `read_archive()`

```
void GiNaC::function::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Construct object from [archive\\_node](#).

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::archive\\_node::find\\_string\(\)](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

### 8.63.3.16 `info()`

```
bool GiNaC::function::info (
    unsigned inf) const [override], [virtual]
```

Implementation of [ex::info](#) for functions.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::info\(\)](#), [GiNaC::function\\_options::info\\_f](#), [GiNaC::function\\_options::info\\_use\\_exvector\\_arg](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

### 8.63.3.17 derivative()

```
ex GiNaC::function::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for functions.

It applies the chain rule, except for the Order term function. @see [ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [expl\\_derivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pderivative\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

### 8.63.3.18 is\_equal\_same\_type()

```
bool GiNaC::function::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< std::vector >](#).

References [function\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::container< std::vector >::is\\_equal\\_same\\_type\(\)](#), and [serial](#).

### 8.63.3.19 match\_same\_type()

```
bool GiNaC::function::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [function\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [serial](#).

**8.63.3.20 return\_type()**

```
unsigned GiNaC::function::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

**8.63.3.21 return\_type\_tinfo()**

```
return_type_t GiNaC::function::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type\\_tinfo](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

**8.63.3.22 pderivative()**

```
ex GiNaC::function::pderivative (
    unsigned diff_param) const [protected]
```

References [current\\_serial](#), [GiNaC::function\\_options::derivative\\_f](#), [GiNaC::function\\_options::derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [derivative\(\)](#), and [GiNaC::fderivative::eval\(\)](#).

**8.63.3.23 expl\_derivative()**

```
ex GiNaC::function::expl_derivative (
    const symbol & s) const [protected]
```

References [current\\_serial](#), [GiNaC::function\\_options::expl\\_derivative\\_f](#), [GiNaC::function\\_options::expl\\_derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [derivative\(\)](#).

**8.63.3.24 registered\_functions()**

```
std::vector< function_options > & GiNaC::function::registered_functions () [static], [protected]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::eval\(\)](#), [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [expl\\_derivative\(\)](#), [find\\_function\(\)](#), [get\\_name\(\)](#), [get\\_registered\\_functions\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [pderivative\(\)](#), [power\(\)](#), [print\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [register\\_new\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

**8.63.3.25 lookup\_remember\_table()**

```
bool GiNaC::function::lookup_remember_table (
    ex & result) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**8.63.3.26 store\_remember\_table()**

```
void GiNaC::function::store_remember_table (
    ex const & result) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**8.63.3.27 power()**

```
ex GiNaC::function::power (
    const ex & exp) const
```

References [current\\_serial](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::function\\_options::power\\_f](#), [GiNaC::function\\_options::power\\_use\\_exvector\\_args](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**8.63.3.28 register\_new()**

```
unsigned GiNaC::function::register_new (
    function_options const & opt) [static]
```

References [GiNaC::function\\_options::functions\\_with\\_same\\_name](#), [GiNaC::function\\_options::name](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::remember\\_assoc\\_size](#), [GiNaC::function\\_options::remember\\_size](#), [GiNaC::function\\_options::remember\\_str](#), [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

**8.63.3.29 find\_function()**

```
unsigned GiNaC::function::find_function (
    const std::string & name,
    unsigned nparams) [static]
```

Find serial number of function by name and number of parameters.

Throws exception if function was not found.

References [registered\\_functions\(\)](#), and [serial](#).

### 8.63.3.30 `get_registered_functions()`

```
std::vector< function\_options > GiNaC::function::get_registered_functions () [inline], [static]
```

References [registered\\_functions\(\)](#).

### 8.63.3.31 `get_serial()`

```
unsigned GiNaC::function::get_serial () const [inline]
```

References [serial](#).

### 8.63.3.32 `get_name()`

```
std::string GiNaC::function::get_name () const
```

Return the print name of the function.

References [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), and [serial](#).

## 8.63.4 Friends And Related Symbol Documentation

### 8.63.4.1 `remember_table_entry`

```
friend class remember_table_entry [friend]
```

References [function\(\)](#), [print\(\)](#), and [remember\\_table\\_entry](#).

Referenced by [remember\\_table\\_entry](#).

## 8.63.5 Member Data Documentation

### 8.63.5.1 `current_serial`

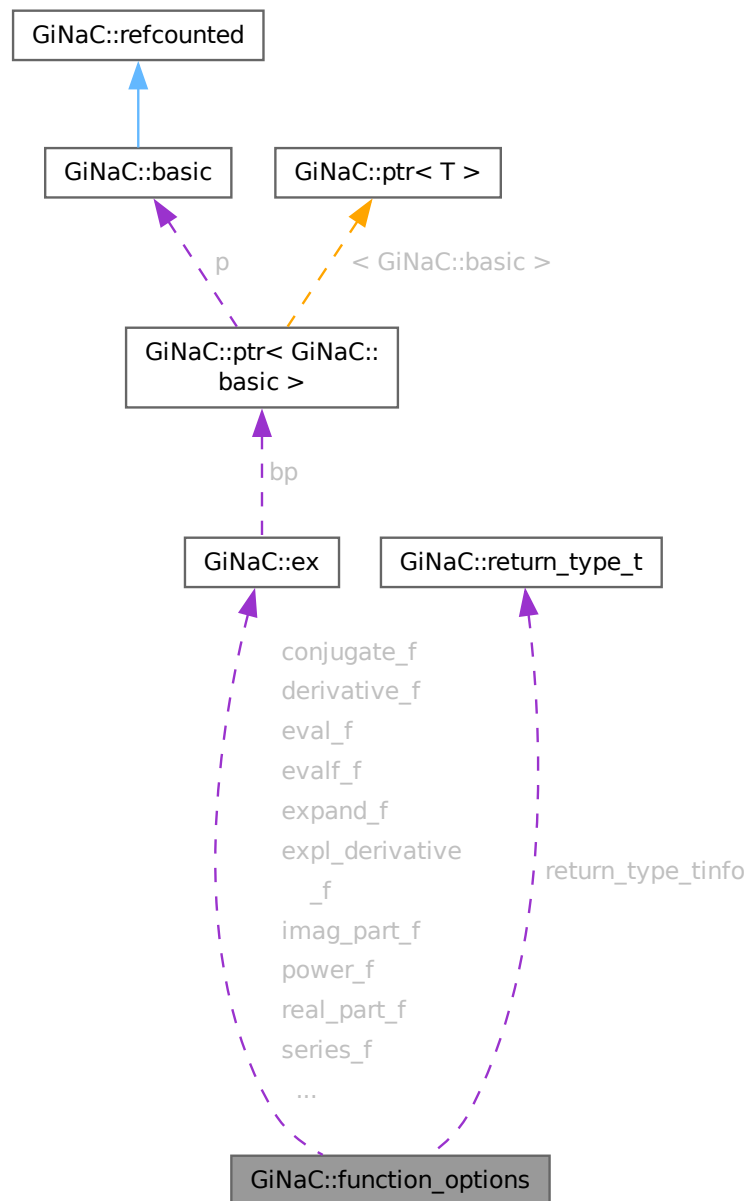
```
unsigned GiNaC::function::current_serial = 0 [static]
```

This can be used as a hook for external applications.

Referenced by [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [expl\\_derivative\(\)](#), [pderivative\(\)](#), [power\(\)](#), [print\(\)](#), and [series\(\)](#).



Collaboration diagram for GiNaC::function\_options:



## Public Member Functions

- [function\\_options](#) ()
- [function\\_options](#) (std::string const &n, std::string const &tn=std::string())
- [function\\_options](#) (std::string const &n, unsigned np)
- [~function\\_options](#) ()
- void [initialize](#) ()
- [function\\_options](#) & [dummy](#) ()
- [function\\_options](#) & [set\\_name](#) (std::string const &n, std::string const &tn=std::string())

- [function\\_options](#) & [latex\\_name](#) (std::string const &tn)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_1 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_2 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_3 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_4 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_5 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_6 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_7 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_8 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_9 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_10 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_11 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_12 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_13 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_14 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_1 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_2 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_3 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_4 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_5 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_6 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_7 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_8 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_9 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_10 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_11 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_12 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_13 e)
- [function\\_options](#) & [evalf\\_func](#) (evalf\_funcp\_14 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_1 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_2 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_3 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_4 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_5 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_6 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_7 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_8 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_9 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_10 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_11 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_12 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_13 e)
- [function\\_options](#) & [conjugate\\_func](#) (conjugate\_funcp\_14 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_1 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_2 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_3 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_4 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_5 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_6 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_7 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_8 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_9 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_10 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_11 e)
- [function\\_options](#) & [real\\_part\\_func](#) (real\_part\_funcp\_12 e)



- Generated by Doxygen

- [function\\_options & expl\\_derivative\\_func \(expl\\_derivative\\_funcp\\_12 e\)](#)
- [function\\_options & expl\\_derivative\\_func \(expl\\_derivative\\_funcp\\_13 e\)](#)
- [function\\_options & expl\\_derivative\\_func \(expl\\_derivative\\_funcp\\_14 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_1 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_2 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_3 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_4 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_5 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_6 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_7 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_8 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_9 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_10 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_11 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_12 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_13 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_14 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_1 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_2 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_3 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_4 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_5 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_6 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_7 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_8 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_9 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_10 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_11 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_12 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_13 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_14 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_1 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_2 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_3 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_4 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_5 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_6 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_7 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_8 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_9 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_10 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_11 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_12 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_13 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_14 e\)](#)
- [function\\_options & eval\\_func \(eval\\_funcp\\_exvector e\)](#)
- [function\\_options & evalf\\_func \(evalf\\_funcp\\_exvector e\)](#)
- [function\\_options & conjugate\\_func \(conjugate\\_funcp\\_exvector e\)](#)
- [function\\_options & real\\_part\\_func \(real\\_part\\_funcp\\_exvector e\)](#)
- [function\\_options & imag\\_part\\_func \(imag\\_part\\_funcp\\_exvector e\)](#)
- [function\\_options & expand\\_func \(expand\\_funcp\\_exvector e\)](#)
- [function\\_options & derivative\\_func \(derivative\\_funcp\\_exvector e\)](#)
- [function\\_options & expl\\_derivative\\_func \(expl\\_derivative\\_funcp\\_exvector e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_exvector e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_exvector e\)](#)

- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_exvector](#) e)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_1](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_2](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_3](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_4](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_5](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_6](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_7](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_8](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_9](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_10](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_11](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_12](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_13](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_14](#) p)
- [template<class Ctx>](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_exvector](#) p)
- [function\\_options](#) & [set\\_return\\_type](#) (unsigned rt, const [return\\_type\\_t](#) \*rtt=nullptr)
- [function\\_options](#) & [do\\_not\\_evalf\\_params](#) ()
- [function\\_options](#) & [remember](#) (unsigned size, unsigned assoc\_size=0, unsigned strategy=[remember\\_strategies::delete\\_never](#))
- [function\\_options](#) & [overloaded](#) (unsigned o)
- [function\\_options](#) & [set\\_symmetry](#) (const [symmetry](#) &s)
- [std::string](#) [get\\_name](#) () const
- unsigned [get\\_nparams](#) () const

### Protected Member Functions

- bool [has\\_derivative](#) () const
- bool [has\\_power](#) () const
- void [test\\_and\\_set\\_nparams](#) (unsigned n)
- void [set\\_print\\_func](#) (unsigned id, [print\\_funcp](#) f)

### Protected Attributes

- [std::string](#) [name](#)
- [std::string](#) [TeX\\_name](#)
- unsigned [nparams](#)
- [eval\\_funcp](#) [eval\\_f](#)
- [evalf\\_funcp](#) [evalf\\_f](#)
- [conjugate\\_funcp](#) [conjugate\\_f](#)



[eval\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#),  
[evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#),  
[expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#),  
[expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#), [expand\\_func\(\)](#),  
[expand\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#),  
[expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#),  
[expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#),  
[expl\\_derivative\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#),  
[imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#),  
[imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#),  
[info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#),  
[info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [latex\\_name\(\)](#), [overloaded\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#),  
[power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#),  
[power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#),  
[print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#),  
[print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#),  
[print\\_func\(\)](#), [print\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#),  
[real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#),  
[real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [real\\_part\\_func\(\)](#), [remember\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#),  
[series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#),  
[series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [set\\_name\(\)](#), [set\\_return\\_type\(\)](#), and  
[set\\_symmetry\(\)](#).

#### 8.64.1.2 function\_options() [2/3]

```
GiNaC::function_options::function_options (
    std::string const & n,
    std::string const & tn = std::string())
```

References [initialize\(\)](#), and [set\\_name\(\)](#).

#### 8.64.1.3 function\_options() [3/3]

```
GiNaC::function_options::function_options (
    std::string const & n,
    unsigned np)
```

References [initialize\(\)](#), [nparams](#), and [set\\_name\(\)](#).

#### 8.64.1.4 ~function\_options()

```
GiNaC::function_options::~~function_options ()
```

### 8.64.2 Member Function Documentation

#### 8.64.2.1 initialize()

```
void GiNaC::function_options::initialize ()
```

References [conjugate\\_f](#), [conjugate\\_use\\_exvector\\_args](#), [derivative\\_f](#), [derivative\\_use\\_exvector\\_args](#), [eval\\_f](#),  
[eval\\_use\\_exvector\\_args](#), [evalf\\_f](#), [evalf\\_params\\_first](#), [evalf\\_use\\_exvector\\_args](#), [expand\\_f](#), [expand\\_use\\_exvector\\_args](#),  
[expl\\_derivative\\_f](#), [expl\\_derivative\\_use\\_exvector\\_args](#), [functions\\_with\\_same\\_name](#), [imag\\_part\\_f](#), [imag\\_part\\_use\\_exvector\\_args](#),  
[info\\_f](#), [info\\_use\\_exvector\\_args](#), [nparams](#), [power\\_f](#), [power\\_use\\_exvector\\_args](#), [print\\_use\\_exvector\\_args](#),  
[real\\_part\\_f](#), [real\\_part\\_use\\_exvector\\_args](#), [series\\_f](#), [series\\_use\\_exvector\\_args](#), [set\\_name\(\)](#), [symtree](#), [use\\_remember](#),  
and [use\\_return\\_type](#).

Referenced by [function\\_options\(\)](#), [function\\_options\(\)](#), and [function\\_options\(\)](#).

#### 8.64.2.2 dummy()

```
function_options & GiNaC::function_options::dummy () [inline]
```

References [function\\_options\(\)](#).

#### 8.64.2.3 set\_name()

```
function_options & GiNaC::function_options::set_name (
    std::string const & n,
    std::string const & tn = std::string())
```

References [function\\_options\(\)](#), [name](#), and [TeX\\_name](#).

Referenced by [function\\_options\(\)](#), [function\\_options\(\)](#), and [initialize\(\)](#).

#### 8.64.2.4 latex\_name()

```
function_options & GiNaC::function_options::latex_name (
    std::string const & tn)
```

References [function\\_options\(\)](#), and [TeX\\_name](#).

#### 8.64.2.5 eval\_func() [1/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_1 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 8.64.2.6 eval\_func() [2/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_2 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 8.64.2.7 eval\_func() [3/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_3 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.8 eval\_func()** [4/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_4 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.9 eval\_func()** [5/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_5 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.10 eval\_func()** [6/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_6 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.11 eval\_func()** [7/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_7 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.12 eval\_func()** [8/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_8 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.13 eval\_func()** [9/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_9 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.14 eval\_func()** [10/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_10 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.15 eval\_func()** [11/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_11 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.16 eval\_func()** [12/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_12 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.17 eval\_func()** [13/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_13 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.18 eval\_func()** [14/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_14 e)
```

References [eval\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.19 evalf\_func()** [1/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_1 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.20 evalf\_func()** [2/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_2 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.21 evalf\_func()** [3/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_3 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).



**8.64.2.22 evalf\_func()** [4/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_4 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.23 evalf\_func()** [5/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_5 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.24 evalf\_func()** [6/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_6 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.25 evalf\_func()** [7/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_7 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.26 evalf\_func()** [8/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_8 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.27 evalf\_func()** [9/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_9 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.28 evalf\_func()** [10/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_10 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.29 evalf\_func()** [11/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_11 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.30 evalf\_func()** [12/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_12 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.31 evalf\_func()** [13/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_13 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.32 evalf\_func()** [14/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_14 e)
```

References [evalf\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.33 conjugate\_func()** [1/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_1 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.34 conjugate\_func()** [2/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_2 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.35 conjugate\_func()** [3/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_3 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.36 conjugate\_func()** [4/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_4 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.37 conjugate\_func()** [5/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_5 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.38 conjugate\_func()** [6/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_6 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.39 conjugate\_func()** [7/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_7 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.40 conjugate\_func()** [8/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_8 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.41 conjugate\_func()** [9/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_9 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.42 conjugate\_func()** [10/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_10 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.43 conjugate\_func()** [11/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_11 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.44 conjugate\_func()** [12/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_12 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.45 conjugate\_func()** [13/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_13 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.46 conjugate\_func()** [14/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_14 e)
```

References [conjugate\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.47 real\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_1 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.48 real\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_2 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.49 real\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_3 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.50 real\_part\_func()** [4/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_4 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.51 real\_part\_func()** [5/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_5 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.52 real\_part\_func()** [6/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_6 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.53 real\_part\_func()** [7/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_7 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.54 real\_part\_func()** [8/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_8 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.55 real\_part\_func()** [9/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_9 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.56 real\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_10 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.57 real\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_11 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.58 real\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_12 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.59 real\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_13 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.60 real\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_14 e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.61 imag\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_1 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.62 imag\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_2 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.63 imag\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_3 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.64 imag\_part\_func() [4/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_4 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.65 imag\_part\_func() [5/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_5 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.66 imag\_part\_func() [6/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_6 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.67 imag\_part\_func() [7/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_7 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.68 imag\_part\_func() [8/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_8 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.69 imag\_part\_func() [9/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_9 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.70 imag\_part\_func() [10/15]**

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_10 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.71 imag\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_11 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.72 imag\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_12 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.73 imag\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_13 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.74 imag\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_14 e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.75 expand\_func()** [1/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_1 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.76 expand\_func()** [2/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_2 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.77 expand\_func()** [3/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_3 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).



**8.64.2.78 expand\_func()** [4/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_4 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.79 expand\_func()** [5/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_5 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.80 expand\_func()** [6/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_6 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.81 expand\_func()** [7/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_7 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.82 expand\_func()** [8/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_8 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.83 expand\_func()** [9/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_9 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.84 expand\_func()** [10/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_10 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.85 expand\_func()** [11/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_11 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.86 expand\_func()** [12/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_12 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.87 expand\_func()** [13/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_13 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.88 expand\_func()** [14/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_14 e)
```

References [expand\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.89 derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_1 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.90 derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_2 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.91 derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_3 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.92 derivative\_func()** [4/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_4 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.93 derivative\_func()** [5/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_5 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.94 derivative\_func()** [6/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_6 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.95 derivative\_func()** [7/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_7 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.96 derivative\_func()** [8/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_8 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.97 derivative\_func()** [9/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_9 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.98 derivative\_func()** [10/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_10 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.99 derivative\_func()** [11/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_11 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.100 derivative\_func()** [12/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_12 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.101 derivative\_func()** [13/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_13 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.102 derivative\_func()** [14/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_14 e)
```

References [derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.103 expl\_derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_1 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.104 expl\_derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_2 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.105 expl\_derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_3 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.106 expl\_derivative\_func() [4/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_4 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.107 expl\_derivative\_func() [5/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_5 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.108 expl\_derivative\_func() [6/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_6 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.109 expl\_derivative\_func() [7/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_7 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.110 expl\_derivative\_func() [8/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_8 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.111 expl\_derivative\_func() [9/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_9 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.112 expl\_derivative\_func() [10/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_10 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.113 expl\_derivative\_func() [11/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_11 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.114 expl\_derivative\_func() [12/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_12 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.115 expl\_derivative\_func() [13/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_13 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.116 expl\_derivative\_func() [14/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_14 e)
```

References [expl\\_derivative\\_f](#), [function\\_options\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.117 power\_func() [1/15]**

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_1 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.118 power\_func() [2/15]**

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_2 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.119 power\_func() [3/15]**

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_3 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.120 power\_func()** [4/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_4 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.121 power\_func()** [5/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_5 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.122 power\_func()** [6/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_6 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.123 power\_func()** [7/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_7 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.124 power\_func()** [8/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_8 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.125 power\_func()** [9/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_9 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.126 power\_func()** [10/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_10 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.127 power\_func()** [11/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_11 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.128 power\_func()** [12/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_12 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.129 power\_func()** [13/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_13 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.130 power\_func()** [14/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_14 e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.131 series\_func()** [1/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_1 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.132 series\_func()** [2/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_2 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.133 series\_func()** [3/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_3 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).



**8.64.2.134 series\_func()** [4/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_4 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.135 series\_func()** [5/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_5 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.136 series\_func()** [6/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_6 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.137 series\_func()** [7/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_7 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.138 series\_func()** [8/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_8 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.139 series\_func()** [9/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_9 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.140 series\_func()** [10/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_10 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.141 series\_func()** [11/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_11 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.142 series\_func()** [12/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_12 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.143 series\_func()** [13/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_13 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.144 series\_func()** [14/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_14 e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.145 info\_func()** [1/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_1 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.146 info\_func()** [2/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_2 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.147 info\_func()** [3/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_3 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.148 info\_func() [4/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_4 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.149 info\_func() [5/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_5 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.150 info\_func() [6/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_6 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.151 info\_func() [7/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_7 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.152 info\_func() [8/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_8 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.153 info\_func() [9/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_9 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.154 info\_func() [10/15]**

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_10 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.155 info\_func()** [11/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_11 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.156 info\_func()** [12/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_12 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.157 info\_func()** [13/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_13 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.158 info\_func()** [14/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_14 e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.159 eval\_func()** [15/15]

```
function_options & GiNaC::function_options::eval_func (  
    eval_funcp_exvector e)
```

References [eval\\_f](#), [eval\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.160 evalf\_func()** [15/15]

```
function_options & GiNaC::function_options::evalf_func (  
    evalf_funcp_exvector e)
```

References [evalf\\_f](#), [evalf\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.161 conjugate\_func()** [15/15]

```
function_options & GiNaC::function_options::conjugate_func (  
    conjugate_funcp_exvector e)
```

References [conjugate\\_f](#), [conjugate\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.162 real\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::real_part_func (  
    real_part_funcp_exvector e)
```

References [function\\_options\(\)](#), [real\\_part\\_f](#), and [real\\_part\\_use\\_exvector\\_args](#).

**8.64.2.163 imag\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::imag_part_func (  
    imag_part_funcp_exvector e)
```

References [function\\_options\(\)](#), [imag\\_part\\_f](#), and [imag\\_part\\_use\\_exvector\\_args](#).

**8.64.2.164 expand\_func()** [15/15]

```
function_options & GiNaC::function_options::expand_func (  
    expand_funcp_exvector e)
```

References [expand\\_f](#), [expand\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.165 derivative\_func()** [15/15]

```
function_options & GiNaC::function_options::derivative_func (  
    derivative_funcp_exvector e)
```

References [derivative\\_f](#), [derivative\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.166 expl\_derivative\_func()** [15/15]

```
function_options & GiNaC::function_options::expl_derivative_func (  
    expl_derivative_funcp_exvector e)
```

References [expl\\_derivative\\_f](#), [expl\\_derivative\\_use\\_exvector\\_args](#), and [function\\_options\(\)](#).

**8.64.2.167 power\_func()** [15/15]

```
function_options & GiNaC::function_options::power_func (  
    power_funcp_exvector e)
```

References [function\\_options\(\)](#), [power\\_f](#), and [power\\_use\\_exvector\\_args](#).

**8.64.2.168 series\_func()** [15/15]

```
function_options & GiNaC::function_options::series_func (  
    series_funcp_exvector e)
```

References [function\\_options\(\)](#), [series\\_f](#), and [series\\_use\\_exvector\\_args](#).

**8.64.2.169 info\_func()** [15/15]

```
function_options & GiNaC::function_options::info_func (  
    info_funcp_exvector e)
```

References [function\\_options\(\)](#), [info\\_f](#), and [info\\_use\\_exvector\\_args](#).

**8.64.2.170 print\_func()** [1/15]

```
template<class Ctx>  
function_options & GiNaC::function_options::print_func (  
    print_funcp_1 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.171 print\_func()** [2/15]

```
template<class Ctx>  
function_options & GiNaC::function_options::print_func (  
    print_funcp_2 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.172 print\_func()** [3/15]

```
template<class Ctx>  
function_options & GiNaC::function_options::print_func (  
    print_funcp_3 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.173 print\_func()** [4/15]

```
template<class Ctx>  
function_options & GiNaC::function_options::print_func (  
    print_funcp_4 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.174 print\_func()** [5/15]

```
template<class Ctx>  
function_options & GiNaC::function_options::print_func (  
    print_funcp_5 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.175 print\_func()** [6/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_6 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.176 print\_func()** [7/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_7 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.177 print\_func()** [8/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_8 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.178 print\_func()** [9/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_9 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.179 print\_func()** [10/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_10 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.180 print\_func()** [11/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_11 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.181 print\_func()** [12/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_12 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.182 print\_func()** [13/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_13 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.183 print\_func()** [14/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_14 p) [inline]
```

References [function\\_options\(\)](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**8.64.2.184 print\_func()** [15/15]

```
template<class Ctx>
function_options & GiNaC::function_options::print_func (
    print_funcp_exvector p) [inline]
```

References [function\\_options\(\)](#), [print\\_use\\_exvector\\_args](#), and [set\\_print\\_func\(\)](#).

**8.64.2.185 set\_return\_type()**

```
function_options & GiNaC::function_options::set_return_type (
    unsigned rt,
    const return_type_t * rtt = nullptr)
```

References [function\\_options\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [return\\_type](#), [return\\_type\\_tinfo](#), and [use\\_return\\_type](#).

**8.64.2.186 do\_not\_evalf\_params()**

```
function_options & GiNaC::function_options::do_not_evalf_params ()
```

References [evalf\\_params\\_first](#), and [function\\_options\(\)](#).



**8.64.2.187 remember()**

```
function_options & GiNaC::function_options::remember (
    unsigned size,
    unsigned assoc_size = 0,
    unsigned strategy = remember_strategies::delete_never)
```

References [function\\_options\(\)](#), [remember\\_assoc\\_size](#), [remember\\_size](#), [remember\\_strategy](#), and [use\\_remember](#).

**8.64.2.188 overloaded()**

```
function_options & GiNaC::function_options::overloaded (
    unsigned o)
```

References [function\\_options\(\)](#), and [functions\\_with\\_same\\_name](#).

**8.64.2.189 set\_symmetry()**

```
function_options & GiNaC::function_options::set_symmetry (
    const symmetry & s)
```

References [function\\_options\(\)](#), and [symtree](#).

**8.64.2.190 get\_name()**

```
std::string GiNaC::function_options::get_name () const [inline]
```

References [name](#).

**8.64.2.191 get\_nparams()**

```
unsigned GiNaC::function_options::get_nparams () const [inline]
```

References [nparams](#).

**8.64.2.192 has\_derivative()**

```
bool GiNaC::function_options::has_derivative () const [inline], [protected]
```

References [derivative\\_f](#).

**8.64.2.193 has\_power()**

```
bool GiNaC::function_options::has_power () const [inline], [protected]
```

References [power\\_f](#).



### 8.64.3.2 fderivative

```
friend class fderivative [friend]
```

References [fderivative](#).

Referenced by [fderivative](#).

## 8.64.4 Member Data Documentation

### 8.64.4.1 name

```
std::string GiNaC::function_options::name [protected]
```

Referenced by [get\\_name\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::register\\_new\(\)](#), [set\\_name\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

### 8.64.4.2 TeX\_name

```
std::string GiNaC::function_options::TeX_name [protected]
```

Referenced by [latex\\_name\(\)](#), [GiNaC::function::print\(\)](#), and [set\\_name\(\)](#).

### 8.64.4.3 nparams

```
unsigned GiNaC::function_options::nparams [protected]
```

Referenced by [GiNaC::function::conjugate\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::function::expl\\_derivative\(\)](#), [function\\_options\(\)](#), [get\\_nparams\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::function::info\(\)](#), [initialize\(\)](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::function::series\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

### 8.64.4.4 eval\_f

```
eval_funcp GiNaC::function_options::eval_f [protected]
```

Referenced by [GiNaC::function::eval\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), [eval\\_func\(\)](#), and [initialize\(\)](#).

### 8.64.4.5 evalf\_f

```
evalf_funcp GiNaC::function_options::evalf_f [protected]
```

Referenced by [GiNaC::function::evalf\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), [evalf\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.6 conjugate\_f

`conjugate_funcp` `GiNaC::function_options::conjugate_f` [protected]

Referenced by `GiNaC::function::conjugate()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, and `initialize()`.

#### 8.64.4.7 real\_part\_f

`real_part_funcp` `GiNaC::function_options::real_part_f` [protected]

Referenced by `initialize()`, `GiNaC::function::real_part()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, and `real_part_func()`.

#### 8.64.4.8 imag\_part\_f

`imag_part_funcp` `GiNaC::function_options::imag_part_f` [protected]

Referenced by `GiNaC::function::imag_part()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, and `initialize()`.

#### 8.64.4.9 expand\_f

`expand_funcp` `GiNaC::function_options::expand_f` [protected]

Referenced by `GiNaC::function::expand()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, and `initialize()`.

#### 8.64.4.10 derivative\_f

`derivative_funcp` `GiNaC::function_options::derivative_f` [protected]

Referenced by `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `has_derivative()`, `initialize()`, and `GiNaC::function::pderivative()`.

#### 8.64.4.11 expl\_derivative\_f

`expl_derivative_funcp` `GiNaC::function_options::expl_derivative_f` [protected]

Referenced by `GiNaC::function::expl_derivative()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, and `initialize()`.

#### 8.64.4.12 power\_f

`power_funcp` GiNaC::function\_options::power\_f [protected]

Referenced by [has\\_power\(\)](#), [initialize\(\)](#), [GiNaC::function::power\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), [power\\_func\(\)](#), and [power\\_func\(\)](#).

#### 8.64.4.13 series\_f

`series_funcp` GiNaC::function\_options::series\_f [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::series\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), [series\\_func\(\)](#), and [series\\_func\(\)](#).

#### 8.64.4.14 print\_dispatch\_table

`std::vector<print_funcp>` GiNaC::function\_options::print\_dispatch\_table [protected]

Referenced by [GiNaC::function::print\(\)](#), and [set\\_print\\_func\(\)](#).

#### 8.64.4.15 info\_f

`info_funcp` GiNaC::function\_options::info\_f [protected]

Referenced by [GiNaC::function::info\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), [info\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.16 evalf\_params\_first

`bool` GiNaC::function\_options::evalf\_params\_first [protected]

Referenced by [do\\_not\\_evalf\\_params\(\)](#), [GiNaC::function::evalf\(\)](#), and [initialize\(\)](#).

#### 8.64.4.17 use\_return\_type

`bool` GiNaC::function\_options::use\_return\_type [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 8.64.4.18 return\_type

`unsigned` GiNaC::function\_options::return\_type [protected]

Referenced by [GiNaC::function::return\\_type\(\)](#), and [set\\_return\\_type\(\)](#).

#### 8.64.4.19 return\_type\_tinfo

`return_type_t` `GiNaC::function_options::return_type_tinfo` [protected]

Referenced by [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 8.64.4.20 use\_remember

`bool` `GiNaC::function_options::use_remember` [protected]

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 8.64.4.21 remember\_size

`unsigned` `GiNaC::function_options::remember_size` [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 8.64.4.22 remember\_assoc\_size

`unsigned` `GiNaC::function_options::remember_assoc_size` [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 8.64.4.23 remember\_strategy

`unsigned` `GiNaC::function_options::remember_strategy` [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 8.64.4.24 eval\_use\_exvector\_args

`bool` `GiNaC::function_options::eval_use_exvector_args` [protected]

Referenced by [GiNaC::function::eval\(\)](#), [eval\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.25 evalf\_use\_exvector\_args

`bool` `GiNaC::function_options::evalf_use_exvector_args` [protected]

Referenced by [GiNaC::function::evalf\(\)](#), [evalf\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.26 conjugate\_use\_exvector\_args

`bool` `GiNaC::function_options::conjugate_use_exvector_args` [protected]

Referenced by [GiNaC::function::conjugate\(\)](#), [conjugate\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.27 real\_part\_use\_exvector\_args

```
bool GiNaC::function_options::real_part_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::real\\_part\(\)](#), and [real\\_part\\_func\(\)](#).

#### 8.64.4.28 imag\_part\_use\_exvector\_args

```
bool GiNaC::function_options::imag_part_use_exvector_args [protected]
```

Referenced by [GiNaC::function::imag\\_part\(\)](#), [imag\\_part\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.29 expand\_use\_exvector\_args

```
bool GiNaC::function_options::expand_use_exvector_args [protected]
```

Referenced by [GiNaC::function::expand\(\)](#), [expand\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.30 derivative\_use\_exvector\_args

```
bool GiNaC::function_options::derivative_use_exvector_args [protected]
```

Referenced by [derivative\\_func\(\)](#), [initialize\(\)](#), and [GiNaC::function::pderivative\(\)](#).

#### 8.64.4.31 expl\_derivative\_use\_exvector\_args

```
bool GiNaC::function_options::expl_derivative_use_exvector_args [protected]
```

Referenced by [GiNaC::function::expl\\_derivative\(\)](#), [expl\\_derivative\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.32 power\_use\_exvector\_args

```
bool GiNaC::function_options::power_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::power\(\)](#), and [power\\_func\(\)](#).

#### 8.64.4.33 series\_use\_exvector\_args

```
bool GiNaC::function_options::series_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::series\(\)](#), and [series\\_func\(\)](#).

#### 8.64.4.34 print\_use\_exvector\_args

```
bool GiNaC::function_options::print_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::print\(\)](#), and [print\\_func\(\)](#).

#### 8.64.4.35 info\_use\_exvector\_args

`bool GiNaC::function_options::info_use_exvector_args [protected]`

Referenced by [GiNaC::function::info\(\)](#), [info\\_func\(\)](#), and [initialize\(\)](#).

#### 8.64.4.36 functions\_with\_same\_name

`unsigned GiNaC::function_options::functions_with_same_name [protected]`

Referenced by [initialize\(\)](#), [overloaded\(\)](#), and [GiNaC::function::register\\_new\(\)](#).

#### 8.64.4.37 symtree

`ex GiNaC::function_options::symtree [protected]`

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), and [set\\_symmetry\(\)](#).

The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 8.65 GiNaC::G2\_SERIAL Class Reference

Generalized multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.65.1 Detailed Description

Generalized multiple polylogarithm.

### 8.65.2 Member Data Documentation

#### 8.65.2.1 serial

`unsigned GiNaC::G2_SERIAL::serial [static]`

#### Initial value:

```
= function::register_new(function_options("G", 2).
    evalf_func(G2_evalf).
    eval_func(G2_eval).
    overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)



## 8.66 GiNaC::G3\_SERIAL Class Reference

Generalized multiple polylogarithm with explicit imaginary parts.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.66.1 Detailed Description

Generalized multiple polylogarithm with explicit imaginary parts.

### 8.66.2 Member Data Documentation

#### 8.66.2.1 serial

```
unsigned GiNaC::G3_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("G", 3).
    evalf_func(G3_evalf).
    eval_func(G3_eval).
    overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 8.67 GiNaC::gcd\_options Struct Reference

Flags to control the behavior of [gcd\(\)](#) and friends.

```
#include <normal.h>
```

### Public Types

- enum { [no\\_heur\\_gcd](#) = 2 , [no\\_part\\_factored](#) = 4 , [use\\_sr\\_gcd](#) = 8 }

### 8.67.1 Detailed Description

Flags to control the behavior of [gcd\(\)](#) and friends.

### 8.67.2 Member Enumeration Documentation

#### 8.67.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<a href="#">no_heur_gcd</a>	Usually <a href="#">GiNaC</a> tries heuristic GCD first, because typically it's much faster than anything else. Even if heuristic algorithm fails, the overhead is negligible w.r.t. cost of computing the GCD by some other method. However, some people dislike it, so here's a flag which tells <a href="#">GiNaC</a> to NOT use the heuristic algorithm.
-----------------------------	--

no_part_factored	<a href="#">GiNaC</a> tries to avoid expanding expressions when computing GCDs. This is a good idea, but some people dislike it. Hence the flag to disable special handling of partially factored polynomials. DON'T SET THIS unless you <i>really</i> know what are you doing!
use_sr_gcd	By default <a href="#">GiNaC</a> uses modular GCD algorithm. Typically it's much faster than PRS (pseudo remainder sequence) algorithm. This flag forces <a href="#">GiNaC</a> to use PRS algorithm

The documentation for this struct was generated from the following file:

- [normal.h](#)

## 8.68 GiNaC::gcdheu\_failed Class Reference

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

### 8.68.1 Detailed Description

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

The documentation for this class was generated from the following file:

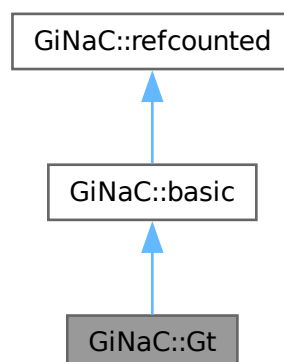
- [normal.cpp](#)

## 8.69 GiNaC::Gt Class Reference

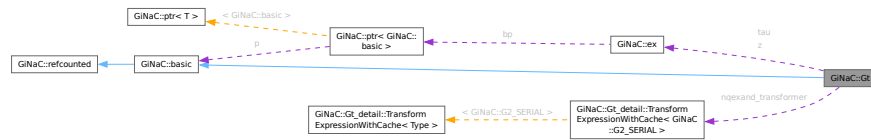
Elliptic Multiple Polylogarithm.

```
#include <Gt.h>
```

Inheritance diagram for GiNaC::Gt:



Collaboration diagram for GiNaC::Gt:



## Classes

- struct [kernel](#)

## Public Member Functions

- [Gt](#) (const [ex](#) &args, const [ex](#) &z, const [ex](#) &tau)
- const [ex](#) & [getTau](#) () const
- const [ex](#) & [getZ](#) () const
- const std::vector< [kernel](#) > & [getArgs](#) () const
- unsigned [precedence](#) () const override
 

*Return relative operator precedence (for parenthezing output).*
- [ex eval](#) () const override
 

*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override
 

*Evaluate object numerically.*
- size\_t [nops](#) () const override
 

*Number of operands/members.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override
 

*Substitute a set of objects by arbitrary expressions.*
- unsigned [calchash](#) () const override
 

*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- bool [has](#) (const [ex](#) &other, unsigned options=0) const override
 

*Test for occurrence of a pattern.*
- bool [match](#) (const [ex](#) &pattern, [exmap](#) &repl\_lst) const override
 

*Check whether the expression matches a given pattern.*
- void [archive](#) ([archive\\_node](#) &n) const override
 

*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override
 

*Read (a.k.a.*
- [ex](#) [zisToFundamental](#) (const [ex](#) \*points=nullptr) const
- [ex](#) [regularise](#) (const [ex](#) \*points=nullptr) const
- [matrix](#) [findMoebiusTransform](#) (const [ex](#) \*points=nullptr) const
- [ex](#) [tauToFundamental](#) (const [ex](#) \*points=nullptr) const
- std::vector< [ex](#) > [decomposeIntegrationPath](#) (const [ex](#) \*points=nullptr) const
- [ex](#) [applyIntegrationPath](#) (const std::vector< [ex](#) > &endpoints) const
- [numeric](#) [qExpand](#) (const [ex](#) \*points=nullptr) const
- [numeric](#) [evaluate](#) (const [ex](#) \*points=nullptr) const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- const `basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position `i`.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position `i`.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object `s`.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object `s`.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree `n` in object `s`.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`

Default implementation of `ex::to_rational()`.

- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

- virtual `numeric max_coefficient () const`

Implementation `ex::max_coefficient()`.

- virtual `exvector get_free_indices () const`

Return a vector containing the free indices of an expression.

- virtual `ex add_indexed (const ex &self, const ex &other) const`

Add two indexed expressions.

- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`

Multiply an indexed expression with a scalar.

- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`

Try to contract two indexed expressions that appear in the same product.

- virtual `unsigned return_type () const`
- virtual `return_type_t return_type_tinfo () const`
- virtual `ex conjugate () const`
- virtual `ex real_part () const`
- virtual `ex imag_part () const`

- `template<class T>`

`void print_dispatch (const print_context &c, unsigned level) const`

Like `print()`, but dispatch to the specified class.

- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`

Like `print()`, but dispatch to the specified class.

- `ex subs_one_level (const exmap &m, unsigned options) const`

Helper function for `subs()`.

- `ex diff (const symbol &s, unsigned nth=1) const`

Default interface of `nth` derivative `ex::diff(s, n)`.

- `int compare (const basic &other) const`

Compare objects syntactically to establish canonical ordering.

- `bool is_equal (const basic &other) const`

Test for syntactic equality.

- `const basic & hold () const`

Stop further evaluation.

- `unsigned gethash () const`

- `const basic & setflag (unsigned f) const`

Set some `status_flags`.

- `const basic & clearflag (unsigned f) const`

Clear some `status_flags`.

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

### Static Public Member Functions

- static `ex ex_zisToFundamental` (const `ex` &expr, const `ex` \*points=nullptr)
- static `ex ex_regularise` (const `ex` &expr, const `ex` \*points=nullptr)
- static `ex ex_tauToFundamental` (const `ex` &expr, const `ex` \*points=nullptr)
- static `ex ex_cutIntegrationPath` (const `ex` &expr, const `ex` \*points=nullptr)
- static `numeric ex_qExpand` (const `ex` &expr, const `ex` \*points=nullptr)
- static `ex ex_prepare` (const `ex` &expr, const `ex` \*points=nullptr)
- static `numeric ex_evaluate` (const `ex` &expr, const `ex` \*points=nullptr)
- static `lst lst_evaluate` (const `lst` &list, const `ex` \*points=nullptr)
- static `ex ex_zisToFundamental` (const `ex` &expr, const `ex` &points)
- static `ex ex_regularise` (const `ex` &expr, const `ex` &points)
- static `ex ex_tauToFundamental` (const `ex` &expr, const `ex` &points)
- static `ex ex_cutIntegrationPath` (const `ex` &expr, const `ex` &points)
- static `numeric ex_qExpand` (const `ex` &expr, const `ex` &points)
- static `ex ex_prepare` (const `ex` &expr, const `ex` &points)
- static `numeric ex_evaluate` (const `ex` &expr, const `ex` &points)
- static `lst lst_evaluate` (const `lst` &list, const `ex` &points)
- static `ex apply_function_recursive` (const `ex` &input, const std::function< `ex`(const `Gt` &)> &func)
- static void `clear_polylog_cache` ()

### Static Public Attributes

- static double `cutIntegrationPath_threshold_real` = 0.4
- static double `cutIntegrationPath_threshold_imag` = 0.4
- static size\_t `qExpand_minOrder` = 4
- static size\_t `qExpand_stepSize` = 4
- static bool `auto_clear_polylog_cache` = false
- static bool `enable_tauToFundamental` = true

### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Private Member Functions

- [Gt](#) (std::vector< [kernel](#) > [args](#), [ex](#) [z](#), [ex](#) [tau](#))
- void [setArgs](#) (const [ex](#) &[args](#))

### Static Private Member Functions

- static [numeric to\\_numeric](#) (const [ex](#) &expr, const [ex](#) \*points=nullptr)
- static long [to\\_integer](#) (const [ex](#) &expr, const bool allow\_negative)

### Private Attributes

- size\_t [nargs](#)
- std::vector< [kernel](#) > [args](#)
- [ex](#) [z](#)
- [ex](#) [tau](#)

### Static Private Attributes

- static [Gt\\_detail::TransformExpressionWithCache](#)< [G2\\_SERIAL](#) > [nqexand\\_transformer](#)

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 8.69.1 Detailed Description

Elliptic Multiple Polylogarithm.

## 8.69.2 Constructor & Destructor Documentation

### 8.69.2.1 [Gt\(\)](#) [1/2]

```
GiNaC::Gt::Gt (
    const ex & args,
    const ex & z,
    const ex & tau)
```

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [nargs](#), [setArgs\(\)](#), [tau](#), and [z](#).

Referenced by [apply\\_function\\_recursive\(\)](#), [applyIntegrationPath\(\)](#), [eval\(\)](#), [ex\\_cutIntegrationPath\(\)](#), [ex\\_qExpand\(\)](#), [ex\\_regularise\(\)](#), [ex\\_tauToFundamental\(\)](#), [ex\\_zisToFundamental\(\)](#), [regularise\(\)](#), [subs\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

### 8.69.2.2 Gt() [2/2]

```
GiNaC::Gt::Gt (
    std::vector< kernel > args,
    ex z,
    ex tau) [private]
```

References [args](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [nargs](#), [tau](#), and [z](#).

## 8.69.3 Member Function Documentation

### 8.69.3.1 setArgs()

```
void GiNaC::Gt::setArgs (
    const ex & args) [private]
```

References [args](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [nargs](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [Gt\(\)](#), and [read\\_archive\(\)](#).

### 8.69.3.2 getTau()

```
const ex & GiNaC::Gt::getTau () const [inline]
```

References [GiNaC::basic::ex](#), and [tau](#).

### 8.69.3.3 getZ()

```
const ex & GiNaC::Gt::getZ () const [inline]
```

References [GiNaC::basic::ex](#), and [z](#).

### 8.69.3.4 getArgs()

```
const std::vector< kernel > & GiNaC::Gt::getArgs () const [inline]
```

References [args](#).

### 8.69.3.5 precedence()

```
unsigned GiNaC::Gt::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).



**8.69.3.6 eval()**

```
ex GiNaC::Gt::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::factorial\(\)](#), [GiNaC::basic::flags](#), [Gt\(\)](#), [GiNaC::basic::hold\(\)](#), [nargs](#), [GiNaC::pow\(\)](#), [tau](#), and [z](#).

**8.69.3.7 evalf()**

```
ex GiNaC::Gt::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [args](#), [GiNaC::dynallocate\(\)](#), [evaluate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_a\(\)](#), [nargs](#), [tau](#), and [z](#).

Referenced by [qExpand\(\)](#).

**8.69.3.8 nops()**

```
size_t GiNaC::Gt::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

**8.69.3.9 subs()**

```
ex GiNaC::Gt::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [args](#), [GiNaC::basic::ex](#), [Gt\(\)](#), [nargs](#), [tau](#), and [z](#).

### 8.69.3.10 calchash()

```
unsigned GiNaC::Gt::calchash () const [override], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::basic::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculator](#), [GiNaC::basic::hashvalue](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), [tau](#), and [z](#).

### 8.69.3.11 has()

```
bool GiNaC::Gt::has (
    const ex & pattern,
    unsigned options = 0) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [args](#), [GiNaC::basic::ex](#), [tau](#), and [z](#).

### 8.69.3.12 match()

```
bool GiNaC::Gt::match (
    const ex & pattern,
    exmap & repl_lst) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::ex\\_to\(\)](#).

### 8.69.3.13 archive()

```
void GiNaC::Gt::archive (
    archive\_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::container< C >::append\(\)](#), [nargs](#), [tau](#), and [z](#).

**8.69.3.14 read\_archive()**

```
void GiNaC::Gt::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [args](#), [GiNaC::basic::ex](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [setArgs\(\)](#), [tau](#), and [z](#).

**8.69.3.15 zisToFundamental()**

```
ex GiNaC::Gt::zisToFundamental (
    const ex * points = nullptr) const
```

References [args](#), [GiNaC::class\\_info< OPT >::first](#), [DEBUG\\_GT](#), [GiNaC::basic::ex](#), [GiNaC::factor\(\)](#), [GiNaC::factorial\(\)](#), [Gt\(\)](#), [GiNaC::l](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::is\\_zero\(\)](#), [nargs](#), [GiNaC::ex::normal\(\)](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::real\(\)](#), [tau](#), [to\\_numeric\(\)](#), and [z](#).

**8.69.3.16 regularise()**

```
ex GiNaC::Gt::regularise (
    const ex * points = nullptr) const
```

References [args](#), [DEBUG\\_GT](#), [GiNaC::basic::ex](#), [GiNaC::factorial\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::get\\_vector\(\)](#), [Gt\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::init\(\)](#), [GiNaC::is\\_zero\(\)](#), [nargs](#), [GiNaC::basic\\_multi\\_iterator< T >::overflow\(\)](#), [GiNaC::pow\(\)](#), [tau](#), [to\\_numeric\(\)](#), and [z](#).

**8.69.3.17 findMoebiusTransform()**

```
matrix GiNaC::Gt::findMoebiusTransform (
    const ex * points = nullptr) const
```

References [GiNaC::abs\(\)](#), [GiNaC::basic::ex](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::matrix::mul\(\)](#), [tau](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), and [to\\_numeric\(\)](#).

Referenced by [tauToFundamental\(\)](#).

**8.69.3.18 tauToFundamental()**

```
ex GiNaC::Gt::tauToFundamental (
    const ex * points = nullptr) const
```

References [args](#), [GiNaC::binomial\(\)](#), [DEBUG\\_GT](#), [evaluate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::factorial\(\)](#), [findMoebiusTransform\(\)](#), [Gt\(\)](#), [GiNaC::l](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::log\(\)](#), [nargs](#), [GiNaC::Gt::kernel::ni](#), [GiNaC::basic::normal\(\)](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [tau](#), [to\\_numeric\(\)](#), and [z](#).

**8.69.3.19 decomposeIntegrationPath()**

```
std::vector< ex > GiNaC::Gt::decomposeIntegrationPath (
    const ex * points = nullptr) const
```

References [args](#), [cutIntegrationPath\\_threshold\\_imag](#), [cutIntegrationPath\\_threshold\\_real](#), [DEBUG\\_GT](#), [GiNaC::basic::diff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::real\(\)](#), [tau](#), [GiNaC::numeric::to\\_double\(\)](#), [to\\_numeric\(\)](#), and [z](#).

**8.69.3.20 applyIntegrationPath()**

```
ex GiNaC::Gt::applyIntegrationPath (
    const std::vector< ex > & endpoints) const
```

References [args](#), [GiNaC::Gt\\_detail::deconcatenate\\_path\(\)](#), [GiNaC::basic::ex](#), [Gt\(\)](#), and [tau](#).

**8.69.3.21 qExpand()**

```
numeric GiNaC::Gt::qExpand (
    const ex * points = nullptr) const
```

References [GiNaC::abs\(\)](#), [args](#), [GiNaC::binomial\(\)](#), [GiNaC::class\\_info< OPT >::first](#), [DEBUG\\_GT](#), [GiNaC::Digits](#), [evalf\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::I](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::log\(\)](#), [nargs](#), [nqexand\\_transformer](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [qExpand\\_minOrder](#), [qExpand\\_stepSize](#), [GiNaC::numeric::real\(\)](#), [GiNaC::real\(\)](#), [tau](#), [GiNaC::numeric::to\\_double\(\)](#), [to\\_numeric\(\)](#), [z](#), and [GiNaC::zeta\(\)](#).

**8.69.3.22 evaluate()**

```
numeric GiNaC::Gt::evaluate (
    const ex * points = nullptr) const
```

References [GiNaC::basic::ex](#), and [ex\\_evaluate\(\)](#).

Referenced by [evalf\(\)](#), and [tauToFundamental\(\)](#).

**8.69.3.23 ex\_zisToFundamental() [1/2]**

```
ex GiNaC::Gt::ex_zisToFundamental (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [apply\\_function\\_recursive\(\)](#), [GiNaC::basic::ex](#), and [Gt\(\)](#).

Referenced by [ex\\_prepare\(\)](#), and [ex\\_zisToFundamental\(\)](#).

**8.69.3.24 `ex_regularise()`** [1/2]

```
ex GiNaC::Gt::ex_regularise (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [apply\\_function\\_recursive\(\)](#), [GiNaC::basic::ex](#), and [Gt\(\)](#).

Referenced by [ex\\_prepare\(\)](#), and [ex\\_regularise\(\)](#).

**8.69.3.25 `ex_tauToFundamental()`** [1/2]

```
ex GiNaC::Gt::ex_tauToFundamental (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [apply\\_function\\_recursive\(\)](#), [GiNaC::basic::ex](#), and [Gt\(\)](#).

Referenced by [ex\\_prepare\(\)](#), and [ex\\_tauToFundamental\(\)](#).

**8.69.3.26 `ex_cutIntegrationPath()`** [1/2]

```
ex GiNaC::Gt::ex_cutIntegrationPath (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [apply\\_function\\_recursive\(\)](#), [GiNaC::basic::ex](#), and [Gt\(\)](#).

Referenced by [ex\\_cutIntegrationPath\(\)](#), and [ex\\_prepare\(\)](#).

**8.69.3.27 `ex_qExpand()`** [1/2]

```
numeric GiNaC::Gt::ex_qExpand (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [apply\\_function\\_recursive\(\)](#), [GiNaC::basic::ex](#), [Gt\(\)](#), and [to\\_numeric\(\)](#).

Referenced by [ex\\_evaluate\(\)](#), [ex\\_qExpand\(\)](#), and [lst\\_evaluate\(\)](#).

**8.69.3.28 `ex_prepare()`** [1/2]

```
ex GiNaC::Gt::ex_prepare (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [DEBUG\\_GT](#), [enable\\_tauToFundamental](#), [GiNaC::basic::ex](#), [ex\\_cutIntegrationPath\(\)](#), [ex\\_regularise\(\)](#), [ex\\_tauToFundamental\(\)](#), and [ex\\_zisToFundamental\(\)](#).

Referenced by [ex\\_evaluate\(\)](#), [ex\\_prepare\(\)](#), and [lst\\_evaluate\(\)](#).

### 8.69.3.29 `ex_evaluate()` [1/2]

```
numeric GiNaC::Gt::ex_evaluate (
    const ex & expr,
    const ex * points = nullptr) [static]
```

References [auto\\_clear\\_polylog\\_cache](#), [clear\\_polylog\\_cache\(\)](#), [DEBUG\\_GT](#), [GiNaC::basic::ex](#), [ex\\_prepare\(\)](#), [ex\\_qExpand\(\)](#), and [to\\_numeric\(\)](#).

Referenced by [evaluate\(\)](#), and [ex\\_evaluate\(\)](#).

### 8.69.3.30 `lst_evaluate()` [1/2]

```
lst GiNaC::Gt::lst_evaluate (
    const lst & list,
    const ex * points = nullptr) [static]
```

References [GiNaC::container< C >::append\(\)](#), [auto\\_clear\\_polylog\\_cache](#), [clear\\_polylog\\_cache\(\)](#), [DEBUG\\_GT](#), [GiNaC::basic::ex](#), [ex\\_prepare\(\)](#), [ex\\_qExpand\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), and [to\\_numeric\(\)](#).

Referenced by [lst\\_evaluate\(\)](#).

### 8.69.3.31 `ex_zisToFundamental()` [2/2]

```
ex GiNaC::Gt::ex_zisToFundamental (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_zisToFundamental\(\)](#).

### 8.69.3.32 `ex_regularise()` [2/2]

```
ex GiNaC::Gt::ex_regularise (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_regularise\(\)](#).

### 8.69.3.33 `ex_tauToFundamental()` [2/2]

```
ex GiNaC::Gt::ex_tauToFundamental (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_tauToFundamental\(\)](#).

**8.69.3.34 ex\_cutIntegrationPath()** [2/2]

```
ex GiNaC::Gt::ex_cutIntegrationPath (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_cutIntegrationPath\(\)](#).

**8.69.3.35 ex\_qExpand()** [2/2]

```
numeric GiNaC::Gt::ex_qExpand (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_qExpand\(\)](#).

**8.69.3.36 ex\_prepare()** [2/2]

```
ex GiNaC::Gt::ex_prepare (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_prepare\(\)](#).

**8.69.3.37 ex\_evaluate()** [2/2]

```
numeric GiNaC::Gt::ex_evaluate (
    const ex & expr,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [ex\\_evaluate\(\)](#).

**8.69.3.38 lst\_evaluate()** [2/2]

```
lst GiNaC::Gt::lst_evaluate (
    const lst & list,
    const ex & points) [static]
```

References [GiNaC::basic::ex](#), and [lst\\_evaluate\(\)](#).

**8.69.3.39 apply\_function\_recursive()**

```
ex GiNaC::Gt::apply_function_recursive (
    const ex & input,
    const std::function< ex(const Gt &)> & func) [static]
```

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), and [Gt\(\)](#).

Referenced by [ex\\_cutIntegrationPath\(\)](#), [ex\\_qExpand\(\)](#), [ex\\_regularise\(\)](#), [ex\\_tauToFundamental\(\)](#), and [ex\\_zisToFundamental\(\)](#).

**8.69.3.40 clear\_polylog\_cache()**

```
void GiNaC::Gt::clear_polylog_cache () [static]
```

References [nqexand\\_transformer](#).

Referenced by [ex\\_evaluate\(\)](#), and [lst\\_evaluate\(\)](#).

**8.69.3.41 do\_print()**

```
void GiNaC::Gt::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [args](#), [nargs](#), [GiNaC::print\\_context::s](#), [tau](#), and [z](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.69.3.42 do\_print\_latex()**

```
void GiNaC::Gt::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

References [args](#), [nargs](#), [GiNaC::print\\_context::s](#), [tau](#), and [z](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.69.3.43 to\_numeric()**

```
numeric GiNaC::Gt::to_numeric (
    const ex & expr,
    const ex * points = nullptr) [static], [private]
```

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [decomposeIntegrationPath\(\)](#), [ex\\_evaluate\(\)](#), [ex\\_qExpand\(\)](#), [findMoebiusTransform\(\)](#), [lst\\_evaluate\(\)](#), [qExpand\(\)](#), [regularise\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

**8.69.3.44 to\_integer()**

```
long GiNaC::Gt::to_integer (
    const ex & expr,
    const bool allow_negative) [static], [private]
```

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [GiNaC::Gt::kernel::kernel\(\)](#).



## 8.69.4 Member Data Documentation

### 8.69.4.1 cutIntegrationPath\_threshold\_real

```
double GiNaC::Gt::cutIntegrationPath_threshold_real = 0.4 [static]
```

Referenced by [decomposeIntegrationPath\(\)](#).

### 8.69.4.2 cutIntegrationPath\_threshold\_imag

```
double GiNaC::Gt::cutIntegrationPath_threshold_imag = 0.4 [static]
```

Referenced by [decomposeIntegrationPath\(\)](#).

### 8.69.4.3 qExpand\_minOrder

```
size_t GiNaC::Gt::qExpand_minOrder = 4 [static]
```

Referenced by [qExpand\(\)](#).

### 8.69.4.4 qExpand\_stepSize

```
size_t GiNaC::Gt::qExpand_stepSize = 4 [static]
```

Referenced by [qExpand\(\)](#).

### 8.69.4.5 auto\_clear\_polylog\_cache

```
bool GiNaC::Gt::auto_clear_polylog_cache = false [static]
```

Referenced by [ex\\_evaluate\(\)](#), and [lst\\_evaluate\(\)](#).

### 8.69.4.6 enable\_tauToFundamental

```
bool GiNaC::Gt::enable_tauToFundamental = true [static]
```

Referenced by [ex\\_prepare\(\)](#).

### 8.69.4.7 nargs

```
size_t GiNaC::Gt::nargs [private]
```

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalf\(\)](#), [Gt\(\)](#), [Gt\(\)](#), [qExpand\(\)](#), [regularise\(\)](#), [setArgs\(\)](#), [subs\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

#### 8.69.4.8 args

```
std::vector<kernel> GiNaC::Gt::args [private]
```

Referenced by [applyIntegrationPath\(\)](#), [calchash\(\)](#), [decomposeIntegrationPath\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalf\(\)](#), [getArgs\(\)](#), [Gt\(\)](#), [has\(\)](#), [qExpand\(\)](#), [read\\_archive\(\)](#), [regularise\(\)](#), [setArgs\(\)](#), [subs\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

#### 8.69.4.9 z

```
ex GiNaC::Gt::z [private]
```

Referenced by [archive\(\)](#), [calchash\(\)](#), [decomposeIntegrationPath\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalf\(\)](#), [getZ\(\)](#), [Gt\(\)](#), [Gt\(\)](#), [has\(\)](#), [qExpand\(\)](#), [read\\_archive\(\)](#), [regularise\(\)](#), [subs\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

#### 8.69.4.10 tau

```
ex GiNaC::Gt::tau [private]
```

Referenced by [applyIntegrationPath\(\)](#), [archive\(\)](#), [calchash\(\)](#), [decomposeIntegrationPath\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalf\(\)](#), [findMoebiusTransform\(\)](#), [getTau\(\)](#), [Gt\(\)](#), [Gt\(\)](#), [has\(\)](#), [qExpand\(\)](#), [read\\_archive\(\)](#), [regularise\(\)](#), [subs\(\)](#), [tauToFundamental\(\)](#), and [zisToFundamental\(\)](#).

#### 8.69.4.11 ngexand\_transformer

```
Gt_detail::TransformExpressionWithCache< G2_SERIAL > GiNaC::Gt::ngexand_transformer [static],
[private]
```

**Initial value:**

```
= [] ()
{
    Digits.add_callback([](long change) -> void {
        if (change>0) Gt::clear_polylog_cache();
    });
    return Gt_detail::TransformExpressionWithCache<G2_SERIAL>{
        [] (const ex& obj) {return obj.evalf();},
        [] (const ex& obj) {return obj.evalf();}
    };
}()
```

Referenced by [clear\\_polylog\\_cache\(\)](#), and [qExpand\(\)](#).

The documentation for this class was generated from the following files:

- [Gt.h](#)
- [Gt.cpp](#)

## 8.70 GiNaC::has\_distance< T > Class Template Reference

SFINAE test for distance.

```
#include <utils_multi_iterator.h>
```

## Public Types

- enum { `value` = sizeof(test<T>(0)) == sizeof(yes\_type) }

## Private Types

- typedef char `yes_type`[1]
- typedef char `no_type`[2]

## Static Private Member Functions

- template<typename C>  
static `yes_type` & `test` (decltype(std::distance< C >))
- template<typename C>  
static `no_type` & `test` (...)

## 8.70.1 Detailed Description

```
template<typename T>
class GiNaC::has_distance< T >
```

SFINAE test for distance.

## 8.70.2 Member Typedef Documentation

### 8.70.2.1 yes\_type

```
template<typename T>
typedef char GiNaC::has_distance< T >::yes_type[1] [private]
```

### 8.70.2.2 no\_type

```
template<typename T>
typedef char GiNaC::has_distance< T >::no_type[2] [private]
```

## 8.70.3 Member Enumeration Documentation

### 8.70.3.1 anonymous enum

```
template<typename T>
anonymous enum
```

#### Enumerator

value	
-------	--

## 8.70.4 Member Function Documentation

### 8.70.4.1 test() [1/2]

```
template<typename T>
template<typename C>
yes_type & GiNaC::has_distance< T >::test (
    decltype(std::distance< C >) ) [static], [private]
```

### 8.70.4.2 test() [2/2]

```
template<typename T>
template<typename C>
no_type & GiNaC::has_distance< T >::test (
    ...) [static], [private]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 8.71 GiNaC::has\_options Class Reference

Flags to control the behavior of [has\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum { [algebraic](#) = 0x0001 }

### 8.71.1 Detailed Description

Flags to control the behavior of [has\(\)](#).

## 8.71.2 Member Enumeration Documentation

### 8.71.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<a href="#">algebraic</a>	enable algebraic matching
---------------------------	---------------------------

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.72 std::hash< GiNaC::ex > Struct Reference

Specialization of std::hash() for ex objects.

```
#include <ex.h>
```

### Public Member Functions

- std::size\_t [operator\(\)](#) (const [GiNaC::ex](#) &e) const noexcept

### 8.72.1 Detailed Description

Specialization of std::hash() for ex objects.

### 8.72.2 Member Function Documentation

#### 8.72.2.1 operator()()

```
std::size_t std::hash< GiNaC::ex >::operator() (
    const GiNaC::ex & e) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

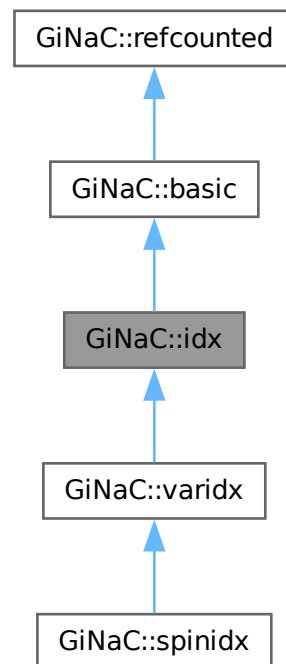
- [ex.h](#)

## 8.73 GiNaC::idx Class Reference

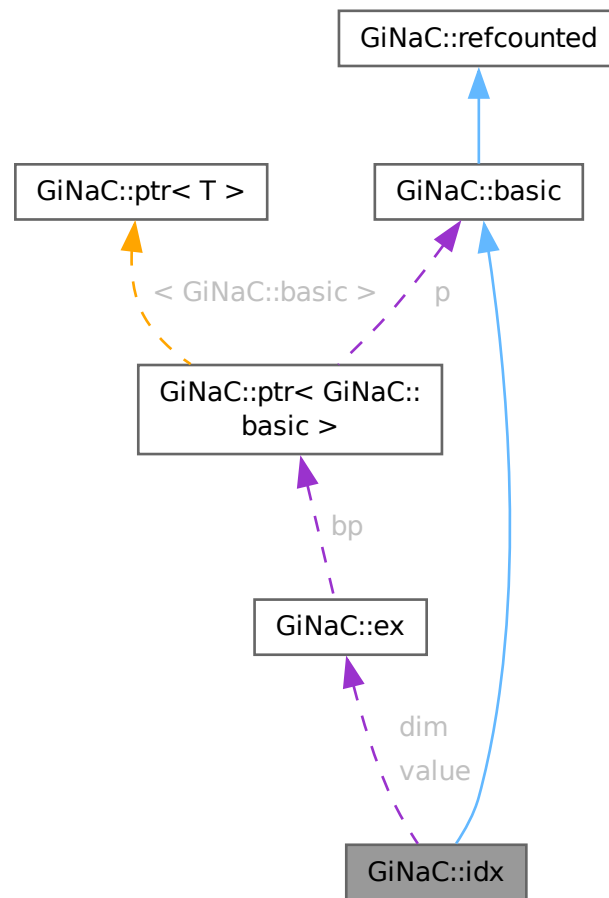
This class holds one index of an indexed object.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::idx:



Collaboration diagram for GiNaC::idx:



## Public Member Functions

- `idx` (const `ex` &`v`, const `ex` &`dim`)  
Construct index with given value and dimension.
- bool `info` (unsigned int) const override  
Information about the object.
- `size_t nops` () const override  
Number of operands/members.
- `ex op` (size\_t `i`) const override  
Return operand/member at position `i`.
- `ex map` (map\_function &`f`) const override  
Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).
- `ex evalf` () const override  
By default, `basic::evalf` would evaluate the index value but we don't want `a.1` to become `a`.
- `ex subs` (const `exmap` &`m`, unsigned options=0) const override  
Substitute a set of objects by arbitrary expressions.

- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Load (deserialize) the object from an archive node.*
- virtual bool `is_dummy_pair_same_type` (const `basic` &other) const  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex` `get_value` () const  
*Get value of index.*
- bool `is_numeric` () const  
*Check whether the index is numeric.*
- bool `is_symbolic` () const  
*Check whether the index is symbolic.*
- `ex` `get_dim` () const  
*Get dimension of index space.*
- bool `is_dim_numeric` () const  
*Check whether the dimension is numeric.*
- bool `is_dim_symbolic` () const  
*Check whether the dimension is symbolic.*
- `ex` `replace_dim` (const `ex` &new\_dim) const  
*Make a new index with the same value but a different dimension.*
- `ex` `minimal_dim` (const `idx` &other) const  
*Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex` `eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex` `evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex` `eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex` `eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual `ex` `operator[]` (const `ex` &index) const
- virtual `ex` `operator[]` (size\_t i) const



- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*

- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an index always returns 0.*
- `bool match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void print_index` (const `print_context` &c, unsigned level) const
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_csrc` (const `print_csrc` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex value](#)  
*Expression that constitutes the index (numeric or symbolic name).*
- [ex dim](#)  
*Dimension of space (can be symbolic or numeric).*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.73.1 Detailed Description

This class holds one index of an indexed object.

Indices can theoretically consist of any symbolic expression but they are usually only just a symbol (e.g. "mu", "i") or numeric (integer). Indices belong to a space with a certain numeric or symbolic dimension.

### 8.73.2 Constructor & Destructor Documentation

#### 8.73.2.1 [idx\(\)](#)

```
GiNaC::idx::idx (
    const ex & v,
    const ex & dim) [explicit]
```

Construct index with given value and dimension.

#### Parameters

<a href="#">v</a>	Value of index (numeric or symbolic)
<a href="#">dim</a>	Dimension of index space (numeric or symbolic)

References [dim](#), [GiNaC::basic::ex](#), [is\\_dim\\_numeric\(\)](#), [GiNaC::info\\_flags::posint](#), and [value](#).

Referenced by [is\\_dummy\\_pair\\_same\\_type\(\)](#), [map\(\)](#), [match\\_same\\_type\(\)](#), [minimal\\_dim\(\)](#), [replace\\_dim\(\)](#), and [subs\(\)](#).

### 8.73.3 Member Function Documentation

#### 8.73.3.1 info()

```
bool GiNaC::idx::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::has\\_indices](#), and [GiNaC::info\\_flags::idx](#).

#### 8.73.3.2 nops()

```
size_t GiNaC::idx::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 8.73.3.3 op()

```
ex GiNaC::idx::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), and [value](#).

#### 8.73.3.4 map()

```
ex GiNaC::idx::map (
    map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::status\\_flags::hash\\_calculated](#), [idx\(\)](#), and [value](#).

### 8.73.3.5 evalf()

```
ex GiNaC::idx::evalf () const [override], [virtual]
```

By default, [basic::evalf](#) would evaluate the index value but we don't want a.1 to become a.

(1.0).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.73.3.6 subs()

```
ex GiNaC::idx::subs (  
    const exmap & m,  
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::find\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [idx\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::subs\\_options::really\\_subs\\_idx](#), and [value](#).

### 8.73.3.7 archive()

```
void GiNaC::idx::archive (  
    archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [dim](#), and [value](#).

### 8.73.3.8 read\_archive()

```
void GiNaC::idx::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [dim](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), and [value](#).

### 8.73.3.9 derivative()

```
ex GiNaC::idx::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an index always returns 0.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and [GiNaC::basic::ex](#).

### 8.73.3.10 match\_same\_type()

```
bool GiNaC::idx::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [GiNaC::basic::basic\(\)](#), [dim](#), [GINAC\\_ASSERT](#), [idx\(\)](#), and [GiNaC::is\\_a\(\)](#).

**8.73.3.11 calchash()**

```
unsigned GiNaC::idx::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.73.3.12 is\_dummy\_pair\_same\_type()**

```
bool GiNaC::idx::is_dummy_pair_same_type (
    const basic & other) const [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented in [GiNaC::spinidx](#), and [GiNaC::varidx](#).

References [GiNaC::basic::basic\(\)](#), [dim](#), [idx\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [value](#).

Referenced by [GiNaC::is\\_dummy\\_pair\(\)](#).

**8.73.3.13 get\_value()**

```
ex GiNaC::idx::get_value () const [inline]
```

Get value of index.

References [GiNaC::basic::ex](#), and [value](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), and [GiNaC::tensdelta::eval\\_indexed\(\)](#).

**8.73.3.14 is\_numeric()**

```
bool GiNaC::idx::is_numeric () const [inline]
```

Check whether the index is numeric.

References [GiNaC::is\\_exactly\\_a\(\)](#), and [value](#).

**8.73.3.15 is\_symbolic()**

```
bool GiNaC::idx::is_symbolic () const [inline]
```

Check whether the index is symbolic.

References [GiNaC::is\\_exactly\\_a\(\)](#), and [value](#).

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**8.73.3.16 get\_dim()**

```
ex GiNaC::idx::get_dim () const [inline]
```

Get dimension of index space.

References [dim](#), and [GiNaC::basic::ex](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**8.73.3.17 is\_dim\_numeric()**

```
bool GiNaC::idx::is_dim_numeric () const [inline]
```

Check whether the dimension is numeric.

References [dim](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

Referenced by [idx\(\)](#).

**8.73.3.18 is\_dim\_symbolic()**

```
bool GiNaC::idx::is_dim_symbolic () const [inline]
```

Check whether the dimension is symbolic.

References [dim](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

**8.73.3.19 replace\_dim()**

```
ex GiNaC::idx::replace_dim (
    const ex & new_dim) const
```

Make a new index with the same value but a different dimension.

References [GiNaC::basic::clearflag\(\)](#), [dim](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::status\\_flags::hash\\_calculated](#), and [idx\(\)](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).



**8.73.3.20 minimal\_dim()**

```
ex GiNaC::idx::minimal_dim (
    const idx & other) const
```

Return the minimum of the dimensions of this and another index.

If this is undecidable, throw an exception.

References [dim](#), [GiNaC::basic::ex](#), [idx\(\)](#), and [GiNaC::minimal\\_dim\(\)](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**8.73.3.21 print\_index()**

```
void GiNaC::idx::print_index (
    const print_context & c,
    unsigned level) const [protected]
```

References [dim](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::print\\_context::options](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [GiNaC::spinidx::do\\_print\\_latex\(\)](#).

**8.73.3.22 do\_print()**

```
void GiNaC::idx::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [print\\_index\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.73.3.23 do\_print\_csrc()**

```
void GiNaC::idx::do_print_csrc (
    const print_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::ex\\_to\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.73.3.24 do\_print\_latex()**

```
void GiNaC::idx::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

References [print\\_index\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.73.3.25 do\_print\_tree()

```
void GiNaC::idx::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.73.4 Member Data Documentation

### 8.73.4.1 value

```
ex GiNaC::idx::value [protected]
```

Expression that constitutes the index (numeric or symbolic name).

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [get\\_value\(\)](#), [idx\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_numeric\(\)](#), [is\\_symbolic\(\)](#), [map\(\)](#), [op\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

### 8.73.4.2 dim

```
ex GiNaC::idx::dim [protected]
```

Dimension of space (can be symbolic or numeric).

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [get\\_dim\(\)](#), [idx\(\)](#), [is\\_dim\\_numeric\(\)](#), [is\\_dim\\_symbolic\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [minimal\\_dim\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [replace\\_dim\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

## 8.74 GiNaC::idx\_is\_equal\_ignore\_dim Struct Reference

### Public Member Functions

- [bool operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

## 8.74.1 Member Function Documentation

### 8.74.1.1 operator>()

```
bool GiNaC::idx_is_equal_ignore_dim::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::ex\\_to\(\)](#), and [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

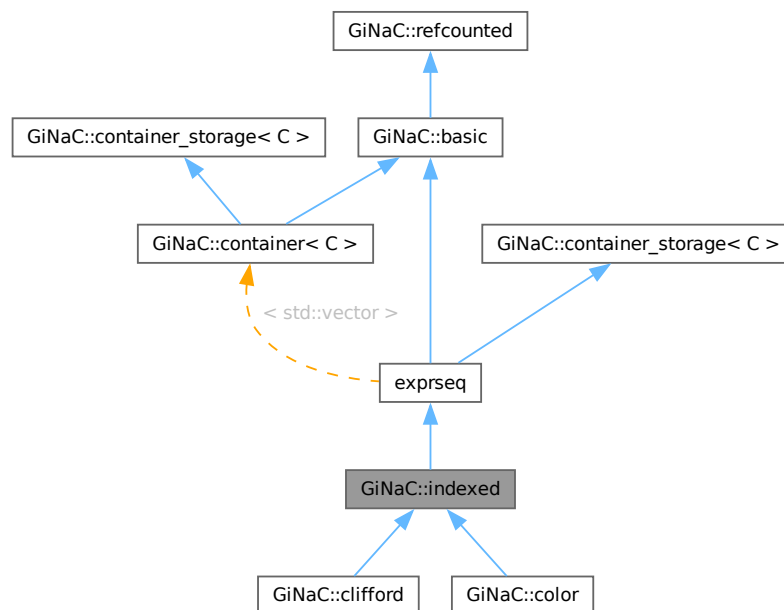
- [indexed.cpp](#)

## 8.75 GiNaC::indexed Class Reference

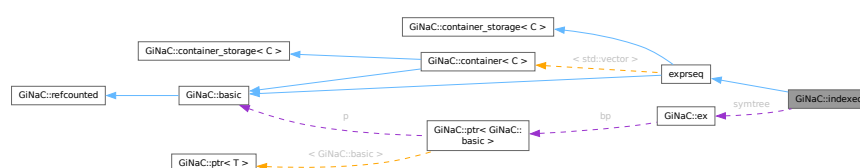
This class holds an indexed expression.

```
#include <indexed.h>
```

Inheritance diagram for GiNaC::indexed:



Collaboration diagram for GiNaC::indexed:



## Public Member Functions

- [indexed](#) (const [ex](#) &b)  
*Construct indexed object with no index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1)  
*Construct indexed object with one index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices and symmetry.*
- [indexed](#) (const [symmetry](#) &symm, const [exprseq](#) &es)
- [indexed](#) (const [symmetry](#) &symm, const [exvector](#) &v)
- [indexed](#) (const [symmetry](#) &symm, [exvector](#) &&v)
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [exvector get\\_free\\_indices](#) () const override  
*Return a vector containing the free indices of an expression.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- bool [all\\_index\\_values\\_are](#) (unsigned inf) const  
*Check whether all index values have a certain property.*
- [exvector get\\_indices](#) () const  
*Return a vector containing the object's indices.*
- [exvector get\\_dummy\\_indices](#) () const  
*Return a vector containing the dummy indices of the object, if any.*
- [exvector get\\_dummy\\_indices](#) (const [indexed](#) &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- bool [has\\_dummy\\_index\\_for](#) (const [ex](#) &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- [ex get\\_symmetry](#) () const  
*Return symmetry properties.*

## Public Member Functions inherited from [GiNaC::container< std::vector >](#)

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [container & prepend](#) (const [ex](#) &b)  
*Add element at front.*
- [container & append](#) (const [ex](#) &b)  
*Add element at back.*
- [container & remove\\_first](#) ()  
*Remove first element.*
- [container & remove\\_last](#) ()  
*Remove last element.*
- [container & remove\\_all](#) ()  
*Remove all elements.*
- [container & sort](#) ()  
*Sort elements.*
- [container & unique](#) ()  
*Remove adjacent duplicate elements.*
- [const\\_iterator begin](#) () const
- [const\\_iterator end](#) () const
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic & operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const

- *Output to stream.*  
virtual void [dbgprint](#) () const
- *Little wrapper around print to be called within a debugger.*  
virtual void [dbgprinttree](#) () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const
- *Test for occurrence of a pattern.*  
virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const
- *Check whether the expression matches a given pattern.*  
virtual [ex map](#) ([map\\_function](#) &f) const
- *Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*  
virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const
- *Check whether this is a polynomial in the given variables.*  
virtual int [degree](#) (const [ex](#) &s) const
- *Return degree of highest power in object s.*  
virtual int [ldegree](#) (const [ex](#) &s) const
- *Return degree of lowest power in object s.*  
virtual [ex coeff](#) (const [ex](#) &s, int n=1) const
- *Return coefficient of degree n in object s.*  
virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const
- *Sort expanded expression in terms of powers of some object(s).*  
virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const
- *Default implementation of [ex::series\(\)](#).*  
virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const
- *Default implementation of [ex::normal\(\)](#).*  
virtual [ex to\\_rational](#) ([exmap](#) &repl) const
- *Default implementation of [ex::to\\_rational\(\)](#).*  
virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*  
virtual [numeric max\\_coefficient](#) () const
- *Implementation [ex::max\\_coefficient\(\)](#).*  
virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const
- *Add two indexed expressions.*  
virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const
- *Multiply an indexed expression with a scalar.*  
virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const
- *Try to contract two indexed expressions that appear in the same product.*  
template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const
- *Like [print\(\)](#), but dispatch to the specified class.*  
void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const
- *Like [print\(\)](#), but dispatch to the specified class.*  
[ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const
- *Helper function for [subs\(\)](#).*

- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&v) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

### Protected Member Functions inherited from `GiNaC::container< std::vector >`

- `ex conjugate` () const override
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned this\_←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT` `subchildren` (const `exmap` &m, unsigned options=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In>  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects).*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)



## Friends

- `ex simplify_indexed` (const `ex` &`e`, `exvector` &`free_indices`, `exvector` &`dummy_indices`, const `scalar_products` &`sp`)  
*Simplify indexed expression, return list of free indices.*
- `ex simplify_indexed_product` (const `ex` &`e`, `exvector` &`free_indices`, `exvector` &`dummy_indices`, const `scalar_products` &`sp`)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- `bool reposition_dummy_indices` (`ex` &`e`, `exvector` &`variant_dummy_indices`, `exvector` &`moved_indices`)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*

## Additional Inherited Members

### Public Types inherited from `GiNaC::container< std::vector >`

- typedef STLT::const\_iterator `const_iterator`
- typedef STLT::const\_reverse\_iterator `const_reverse_iterator`

### Protected Types inherited from `GiNaC::container< std::vector >`

- typedef `container_storage< std::vector >::STLT` `STLT`

### Protected Types inherited from `GiNaC::container_storage< C >`

- typedef `C< ex > STLT`

### Static Protected Member Functions inherited from `GiNaC::container< std::vector >`

- static unsigned `get_default_flags` ()
- static char `get_open_delim` ()
- static char `get_close_delim` ()

### Static Protected Member Functions inherited from `GiNaC::container_storage< C >`

- static void `reserve` (`STLT` &, `size_t`)

## 8.75.1 Detailed Description

This class holds an indexed expression.

It consists of a 'base' expression (the expression being indexed) which can be accessed as `op(0)`, and `n` (`n >= 0`) indices (all of class `idx`), accessible as `op(1)..op(n)`.

## 8.75.2 Constructor & Destructor Documentation

### 8.75.2.1 indexed() [1/13]

```
GiNaC::indexed::indexed (
    const ex & b)
```

Construct indexed object with no index.

#### Parameters

<i>b</i>	Base expression
----------	-----------------

References [indexed\(\)](#), [GiNaC::not\\_symmetric\(\)](#), [symtree](#), and [validate\(\)](#).

Referenced by [eval\(\)](#), [GiNaC::clifford::eval\\_ncmul\(\)](#), [get\\_dummy\\_indices\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [indexed\(\)](#), [simplify\\_indexed](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

### 8.75.2.2 indexed() [2/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1)
```

Construct indexed object with one index.

The index must be of class idx.

#### Parameters

<i>b</i>	Base expression
<i>i1</i>	The index

References [GiNaC::not\\_symmetric\(\)](#), [symtree](#), and [validate\(\)](#).

### 8.75.2.3 indexed() [3/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2)
```

Construct indexed object with two indices.

The indices must be of class idx.

#### Parameters

<i>b</i>	Base expression
----------	-----------------

<i>i1</i>	First index
<i>i2</i>	Second index

References [GiNaC::not\\_symmetric\(\)](#), [symtree](#), and [validate\(\)](#).

#### 8.75.2.4 indexed() [4/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3)
```

Construct indexed object with three indices.

The indices must be of class idx.

##### Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

References [GiNaC::not\\_symmetric\(\)](#), [symtree](#), and [validate\(\)](#).

#### 8.75.2.5 indexed() [5/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4)
```

Construct indexed object with four indices.

The indices must be of class idx.

##### Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

References [GiNaC::not\\_symmetric\(\)](#), [symtree](#), and [validate\(\)](#).

**8.75.2.6 indexed()** [6/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2)
```

Construct indexed object with two indices and a specified symmetry.

The indices must be of class `idx`.

**Parameters**

<i>b</i>	Base expression
<a href="#">symm</a>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index

References [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

**8.75.2.7 indexed()** [7/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2,
    const ex & i3)
```

Construct indexed object with three indices and a specified symmetry.

The indices must be of class `idx`.

**Parameters**

<i>b</i>	Base expression
<a href="#">symm</a>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

References [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

### 8.75.2.8 indexed() [8/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4)
```

Construct indexed object with four indices and a specified symmetry.

The indices must be of class `idx`.

#### Parameters

<i>b</i>	Base expression
----------	-----------------

<a href="#"><i>symm</i></a>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

References [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

#### 8.75.2.9 indexed() [9/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const exvector & iv)
```

Construct indexed object with a specified vector of indices.

The indices must be of class `idx`.

##### Parameters

<i>b</i>	Base expression
<i>iv</i>	Vector of indices

References [GiNaC::not\\_symmetric\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [validate\(\)](#).

#### 8.75.2.10 indexed() [10/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const exvector & iv)
```

Construct indexed object with a specified vector of indices and symmetry.

The indices must be of class `idx`.

##### Parameters

<i>b</i>	Base expression
<a href="#"><i>symm</i></a>	Symmetry of indices
<i>iv</i>	Vector of indices

References [GiNaC::container\\_storage< C >::seq](#), [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

**8.75.2.11 indexed()** [11/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exprseq & es)
```

References [GiNaC::symm\(\)](#), and [symtree](#).

**8.75.2.12 indexed()** [12/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exvector & v)
```

References [GiNaC::symm\(\)](#), and [symtree](#).

**8.75.2.13 indexed()** [13/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    exvector && v)
```

References [GiNaC::symm\(\)](#), and [symtree](#).

**8.75.3 Member Function Documentation****8.75.3.1 precedence()**

```
unsigned GiNaC::indexed::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< std::vector >](#).

Referenced by [print\\_indexed\(\)](#).

**8.75.3.2 info()**

```
bool GiNaC::indexed::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::indexed](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 8.75.3.3 eval()

```
ex GiNaC::indexed::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [indexed\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [thiscontainer\(\)](#).

### 8.75.3.4 real\_part()

```
ex GiNaC::indexed::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< std::vector >](#).

References [info\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 8.75.3.5 imag\_part()

```
ex GiNaC::indexed::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< std::vector >](#).

References [info\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 8.75.3.6 get\_free\_indices()

```
exvector GiNaC::indexed::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [get\\_dummy\\_indices\(\)](#).

### 8.75.3.7 archive()

```
void GiNaC::indexed::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) indexed object into archive.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), and [symtree](#).



### 8.75.3.8 read\_archive()

```
void GiNaC::indexed::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) indexed object from archive.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::ex\\_to\(\)](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [GiNaC::not\\_symmetric\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_symm\(\)](#), [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

### 8.75.3.9 derivative()

```
ex GiNaC::indexed::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an indexed object always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

### 8.75.3.10 thiscontainer() [1/2]

```
ex GiNaC::indexed::thiscontainer (
    const exvector & v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::ex\\_to\(\)](#), [indexed\(\)](#), and [symtree](#).

Referenced by [eval\(\)](#), and [expand\(\)](#).

### 8.75.3.11 thiscontainer() [2/2]

```
ex GiNaC::indexed::thiscontainer (
    exvector && v) const [override], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::ex\\_to\(\)](#), [indexed\(\)](#), and [symtree](#).

### 8.75.3.12 `return_type()`

```
unsigned GiNaC::indexed::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::is\\_a\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

### 8.75.3.13 `return_type_tinfo()`

```
return_type_t GiNaC::indexed::return_type_tinfo () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< std::vector >::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

### 8.75.3.14 `expand()`

```
ex GiNaC::indexed::expand (
    unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_in](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [thiscontainer\(\)](#).

### 8.75.3.15 `all_index_values_are()`

```
bool GiNaC::indexed::all_index_values_are (
    unsigned inf) const
```

Check whether all index values have a certain property.

See also

class [info\\_flags](#)

References [GiNaC::container\\_storage< C >::seq](#).

### 8.75.3.16 `get_indices()`

```
exvector GiNaC::indexed::get_indices () const
```

Return a vector containing the object's indices.

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.75.3.17 get\_dummy\_indices()** [1/2]

```
exvector GiNaC::indexed::get_dummy_indices () const
```

Return a vector containing the dummy indices of the object, if any.

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.75.3.18 get\_dummy\_indices()** [2/2]

```
exvector GiNaC::indexed::get_dummy_indices (
    const indexed & other) const
```

Return a vector containing the dummy indices in the contraction with another indexed object.

This is symmetric: `a.get_dummy_indices(b) == b.get_dummy_indices(a)`

References [GiNaC::find\\_dummy\\_indices\(\)](#), [get\\_free\\_indices\(\)](#), and [indexed\(\)](#).

**8.75.3.19 has\_dummy\_index\_for()**

```
bool GiNaC::indexed::has_dummy_index_for (
    const ex & i) const
```

Check whether the object has an index that forms a dummy index pair with a given index.

References [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.75.3.20 get\_symmetry()**

```
ex GiNaC::indexed::get_symmetry () const [inline]
```

Return symmetry properties.

References [symtree](#).

Referenced by [GiNaC::clifford::get\\_metric\(\)](#).

**8.75.3.21 printindices()**

```
void GiNaC::indexed::printindices (
    const print_context & c,
    unsigned level) const [protected]
```

References [GiNaC::class\\_info< OPT >::first](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), and [print\\_indexed\(\)](#).

**8.75.3.22 print\_indexed()**

```
void GiNaC::indexed::print_indexed (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    unsigned level) const [protected]
```

References [precedence\(\)](#), [printindices\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**8.75.3.23 do\_print()**

```
void GiNaC::indexed::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [print\\_indexed\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.75.3.24 do\_print\_latex()**

```
void GiNaC::indexed::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [print\\_indexed\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.75.3.25 do\_print\_tree()**

```
void GiNaC::indexed::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [printindices\(\)](#), [GiNaC::print\\_context::s](#), [GiNaC::container\\_storage< C >::seq](#), and [symtree](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.75.3.26 validate()**

```
void GiNaC::indexed::validate () const [protected]
```

Check whether all indices are of class `idx` and validate the symmetry tree.

This function is used internally to make sure that all constructed indexed objects really carry indices and not some other classes.

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [validate\(\)](#).

Referenced by [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [read\\_archive\(\)](#), and [validate\(\)](#).

## 8.75.4 Friends And Related Symbol Documentation

### 8.75.4.1 simplify\_indexed

```
ex simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp) [friend]
```

Simplify indexed expression, return list of free indices.

References [GiNaC::\\_ex2](#), [GiNaC::class\\_info< OPT >::first](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::find\\_variant\\_indices\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::idx\\_symmetrization\(\)](#), [indexed\(\)](#), [GiNaC::indices\\_consistent\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::reposition\\_dummy\\_indices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::simplify\\_indexed\(\)](#), and [GiNaC::simplify\\_indexed\\_product\(\)](#).

Referenced by [GiNaC::clifford::get\\_metric\(\)](#), and [GiNaC::clifford::same\\_metric\(\)](#).

### 8.75.4.2 simplify\_indexed\_product

```
ex simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp) [friend]
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::return\\_types::commutative](#), [GiNaC::scalar\\_products::evaluate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::find\\_variant\\_indices\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::idx\\_symmetrization\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::scalar\\_products::is\\_defined\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::minimal\\_dim\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::reposition\\_dummy\\_indices\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), and [GiNaC::simplify\\_indexed\\_product\(\)](#).

### 8.75.4.3 reposition\_dummy\_indices

```
bool reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices) [friend]
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

#### Parameters

<i>e</i>	Object to work on
<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)

<code>moved_indices</code>	The set of indices that have been repositioned (will be changed by this function)
----------------------------	---

#### Returns

true if 'e' was changed

References [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::ex::subs\(\)](#).

## 8.75.5 Member Data Documentation

### 8.75.5.1 symtree

```
ex GiNaC::indexed::symtree [protected]
```

Index symmetry (tree of symmetry objects).

Referenced by [archive\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [get\\_symmetry\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), [thiscontainer\(\)](#), and [validate\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 8.76 GiNaC::info\_flags Class Reference

Possible attributes an object can have.

```
#include <flags.h>
```

### Public Types

- enum {  
[numeric](#) , [real](#) , [rational](#) , [integer](#) ,  
[crational](#) , [cinteger](#) , [positive](#) , [negative](#) ,  
[nonnegative](#) , [posint](#) , [negint](#) , [nonnegint](#) ,  
[even](#) , [odd](#) , [prime](#) , [relation](#) ,  
[relation\\_equal](#) , [relation\\_not\\_equal](#) , [relation\\_less](#) , [relation\\_less\\_or\\_equal](#) ,  
[relation\\_greater](#) , [relation\\_greater\\_or\\_equal](#) , [symbol](#) , [list](#) ,  
[exprseq](#) , [polynomial](#) , [integer\\_polynomial](#) , [cinteger\\_polynomial](#) ,  
[rational\\_polynomial](#) , [crational\\_polynomial](#) , [rational\\_function](#) , [indexed](#) ,  
[has\\_indices](#) , [idx](#) , [expanded](#) , [indefinite](#) }

### 8.76.1 Detailed Description

Possible attributes an object can have.

## 8.76.2 Member Enumeration Documentation

### 8.76.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<code>numeric</code>	
----------------------	--

real	
rational	
integer	
crational	
cinteger	
positive	
negative	
nonnegative	
posint	
negint	
nonnegint	
even	
odd	
prime	
relation	
relation_equal	
relation_not_equal	
relation_less	
relation_less_or_equal	
relation_greater	
relation_greater_or_equal	
symbol	
list	
exprseq	
polynomial	
integer_polynomial	
cinteger_polynomial	
rational_polynomial	
crational_polynomial	
rational_function	
indexed	
has_indices	
idx	
expanded	
indefinite	

The documentation for this class was generated from the following file:

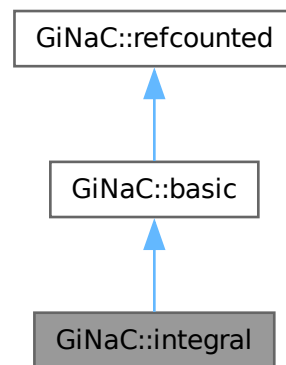
- [flags.h](#)

## 8.77 GiNaC::integral Class Reference

Symbolic integral.

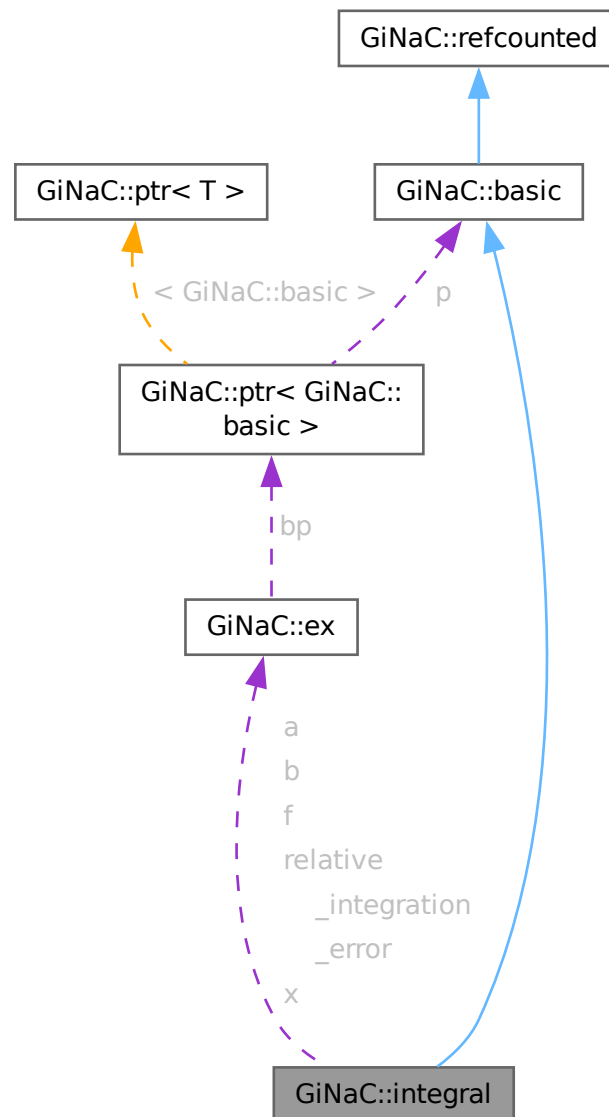
```
#include <integral.h>
```

Inheritance diagram for GiNaC::integral:





Collaboration diagram for GiNaC::integral:



## Public Member Functions

- `integral` (const `ex` &`x_`, const `ex` &`a_`, const `ex` &`b_`, const `ex` &`f_`)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- `ex eval` () const override  
Perform automatic non-interruptive term rewriting rules.
- `ex evalf` () const override  
Evaluate object numerically.
- int `degree` (const `ex` &`s`) const override

- Return degree of highest power in object s.*

  - `int ldegree (const ex &s)` const override
- Return degree of lowest power in object s.*

  - `ex eval_ncmul (const exvector &v)` const override
- Number of operands/members.*

  - `size_t nops ()` const override
- Return operand/member at position i.*

  - `ex op (size_t i)` const override
- Return modifiable operand/member at position i.*

  - `ex & let_op (size_t i)` override
- Expand expression, i.e.*

  - `ex expand (unsigned options=0)` const override
- Return a vector containing the free indices of an expression.*

  - `exvector get_free_indices ()` const override
- Return a vector containing the free indices of an expression.*

  - `unsigned return_type ()` const override
- Return a vector containing the free indices of an expression.*

  - `return_type_t return_type_tinfo ()` const override
- Evaluate integrals, if result is known.*

  - `ex conjugate ()` const override
- Evaluate integrals, if result is known.*

  - `ex eval_integ ()` const override
- Save (a.k.a.)*

  - `void archive (archive_node &n)` const override
- Read (a.k.a.)*

  - `void read_archive (const archive_node &n, lst &syms)` override

## Public Member Functions inherited from GiNaC::basic

- basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*

  - `virtual ~basic ()`
- basic assignment operator: the other object might be of a derived class.*

  - `basic (const basic &other)`
  - `const basic & operator= (const basic &other)`
- Create a clone of this object on the heap.*

  - `virtual basic * duplicate ()` const
- Evaluate sums, products and integer powers of matrices.*

  - `virtual ex evalm ()` const
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

  - `virtual ex eval_indexed (const basic &i)` const
- Output to stream.*

  - `virtual void print (const print_context &c, unsigned level=0)` const
- Little wrapper around print to be called within a debugger.*

  - `virtual void dbgprint ()` const
- Little wrapper around printtree to be called within a debugger.*

  - `virtual void dbgprinttree ()` const
- Information about the object.*

  - `virtual bool info (unsigned inf)` const
- Test for occurrence of a pattern.*

  - `virtual bool has (const ex &other, unsigned options=0)` const

- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Static Public Attributes

- static int [max\\_integration\\_level](#) = 15
- static [ex](#) [relative\\_integration\\_error](#) = 1e-8

## Protected Member Functions

- [ex](#) [derivative](#) (const [symbol](#) &s) const override  
*Default implementation of [ex::diff\(\)](#).*
- [ex](#) [series](#) (const [relational](#) &r, int order, unsigned options=0) const override  
*Default implementation of [ex::series\(\)](#).*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- [ex](#) x
- [ex](#) a
- [ex](#) b
- [ex](#) f

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 8.77.1 Detailed Description

Symbolic integral.

## 8.77.2 Constructor & Destructor Documentation

### 8.77.2.1 `integral()`

```
GiNaC::integral::integral (
    const ex & x_,
    const ex & a_,
    const ex & b_,
    const ex & f_)
```

References [a](#), [b](#), [GiNaC::basic::ex](#), [f](#), [integral\(\)](#), [GiNaC::is\\_a\(\)](#), and [x](#).

Referenced by [derivative\(\)](#), [expand\(\)](#), [integral\(\)](#), and [series\(\)](#).

## 8.77.3 Member Function Documentation

### 8.77.3.1 `precedence()`

```
unsigned GiNaC::integral::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_latex\(\)](#).

### 8.77.3.2 `eval()`

```
ex GiNaC::integral::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [a](#), [b](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::haswild\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**8.77.3.3 evalf()**

```
ex GiNaC::integral::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::adaptivesimpson\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::basic::ex](#), [f](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**8.77.3.4 degree()**

```
int GiNaC::integral::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::degree\(\)](#), [GiNaC::basic::ex](#), and [f](#).

**8.77.3.5 ldegree()**

```
int GiNaC::integral::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ex](#), and [f](#).

**8.77.3.6 eval\_ncmul()**

```
ex GiNaC::integral::eval_ncmul (
    const exvector & v) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [f](#).

**8.77.3.7 nops()**

```
size_t GiNaC::integral::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.77.3.8 op()

```
ex GiNaC::integral::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ex](#), [f](#), [GINAC\\_ASSERT](#), and [x](#).

### 8.77.3.9 let\_op()

```
ex & GiNaC::integral::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [f](#), and [x](#).

### 8.77.3.10 expand()

```
ex GiNaC::integral::expand (
    unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [integral\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

Referenced by [eval\\_integ\(\)](#).

### 8.77.3.11 get\_free\_indices()

```
exvector GiNaC::integral::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), and [f](#).

**8.77.3.12 return\_type()**

```
unsigned GiNaC::integral::return_type () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#).

**8.77.3.13 return\_type\_tinfo()**

```
return_type_t GiNaC::integral::return_type_tinfo () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#).

**8.77.3.14 conjugate()**

```
ex GiNaC::integral::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::dynamlocate\(\)](#), [GiNaC::basic::ex](#), [f](#), and [x](#).

**8.77.3.15 eval\_integ()**

```
ex GiNaC::integral::eval_integ () const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::eval\\_integ\(\)](#), [GiNaC::basic::ex](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::is\\_a\(\)](#), [GiNaC::log\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**8.77.3.16 archive()**

```
void GiNaC::integral::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::archive\\_node::add\\_ex\(\)](#), [b](#), [f](#), and [x](#).



### 8.77.3.17 read\_archive()

```
void GiNaC::integral::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), and [x](#).

### 8.77.3.18 derivative()

```
ex GiNaC::integral::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ex](#), [f](#), [integral\(\)](#), and [x](#).

### 8.77.3.19 series()

```
ex GiNaC::integral::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [protected], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [a](#), [b](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [f](#), [integral\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

### 8.77.3.20 do\_print()

```
void GiNaC::integral::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [a](#), [b](#), [f](#), [GiNaC::print\\_context::s](#), and [x](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.77.3.21 do\_print\_latex()

```
void GiNaC::integral::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [a](#), [b](#), [GiNaC::ex\\_to\(\)](#), [f](#), [precedence\(\)](#), [GiNaC::print\\_context::s](#), and [x](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.77.4 Member Data Documentation

### 8.77.4.1 max\_integration\_level

```
int GiNaC::integral::max_integration_level = 15 [static]
```

Referenced by [GiNaC::adaptivesimpson\(\)](#).

### 8.77.4.2 relative\_integration\_error

```
ex GiNaC::integral::relative_integration_error = 1e-8 [static]
```

Referenced by [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER\(\)](#).

### 8.77.4.3 x

```
ex GiNaC::integral::x [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [integral\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

### 8.77.4.4 a

```
ex GiNaC::integral::a [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [integral\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

## 8.77.4.5 b

```
ex GiNaC::integral::b [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [integral\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

## 8.77.4.6 f

```
ex GiNaC::integral::f [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [integral\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

The documentation for this class was generated from the following files:

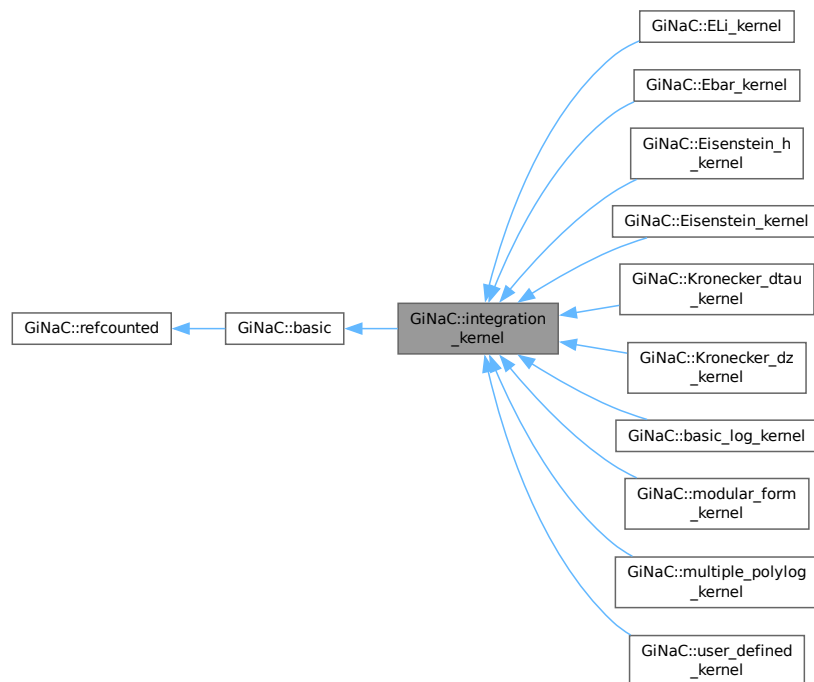
- [integral.h](#)
- [indexed.cpp](#)
- [integral.cpp](#)
- [pseries.cpp](#)

## 8.78 GiNaC::integration\_kernel Class Reference

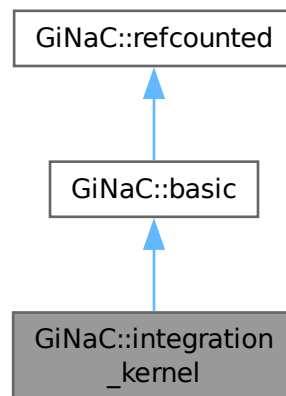
The base class for integration kernels for iterated integrals.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::integration\_kernel:



Collaboration diagram for `GiNaC::integration_kernel`:



### Public Member Functions

- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int i) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const

- Create a clone of this object on the heap.*
- virtual `ex eval` () const
- Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const
- Evaluate object numerically.*
- virtual `ex evalm` () const
- Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const
- Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const
- Output to stream.*
- virtual void `dbgprint` () const
- Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const
- Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const
- Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const
- Information about the object.*
- virtual size\_t `nops` () const
- Number of operands/members.*
- virtual `ex op` (size\_t i) const
- Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)
- Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const
- Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

- Sort expanded expression in terms of powers of some object(s).*

    - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
  - virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
  - template<class T>
  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method Laurent\_series needs to be implemented).*
- virtual `cln::cl_N` [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex` [get\\_numerical\\_value\\_impl](#) (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from GiNaC::basic

- [basic](#) ()
- virtual `ex` [eval\\_ncmul](#) (const `exvector` &v) const
- virtual bool [match\\_same\\_type](#) (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex` [derivative](#) (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int [compare\\_same\\_type](#) (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 8.78.1 Detailed Description

The base class for integration kernels for iterated integrals.

This class represents the differential one-form

$$\omega = d\lambda$$

The integration variable is a dummy variable and does not need to be specified.

### 8.78.2 Member Function Documentation

#### 8.78.2.1 [series\(\)](#)

```
ex GiNaC::integration_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::modular\\_form\\_kernel](#).

References [GiNaC::basic::ex](#), [Laurent\\_series\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::relational::rhs\(\)](#), and [series\(\)](#).

Referenced by [series\(\)](#).

#### 8.78.2.2 [has\\_trailing\\_zero\(\)](#)

```
bool GiNaC::integration_kernel::has_trailing_zero (
    void ) const [virtual]
```

This routine returns true, if the integration kernel has a trailing zero.

References [has\\_trailing\\_zero\(\)](#), and [series\\_coeff\(\)](#).

Referenced by [has\\_trailing\\_zero\(\)](#).



### 8.78.2.3 is\_numeric()

```
bool GiNaC::integration_kernel::is_numeric (
    void ) const [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented in [GiNaC::Ebar\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

References [is\\_numeric\(\)](#).

Referenced by [is\\_numeric\(\)](#).

### 8.78.2.4 Laurent\_series()

```
ex GiNaC::integration_kernel::Laurent_series (
    const ex & x,
    int order) const [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented in [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

References [GiNaC::basic::ex](#), [Laurent\\_series\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::series\(\)](#), and [series\\_coeff\(\)](#).

Referenced by [Laurent\\_series\(\)](#), [series\(\)](#), and [series\\_coeff\(\)](#).

### 8.78.2.5 get\_numerical\_value()

```
ex GiNaC::integration_kernel::get_numerical_value (
    const ex & lambda,
    int N_trunc = 0) const [virtual]
```

Evaluates the integrand at lambda.

Reimplemented in [GiNaC::Ebar\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), and [GiNaC::modular\\_form\\_kernel](#).

References [GiNaC::basic::ex](#), [get\\_numerical\\_value\(\)](#), and [get\\_numerical\\_value\\_impl\(\)](#).

Referenced by [get\\_numerical\\_value\(\)](#).

### 8.78.2.6 uses\_Laurent\_series()

```
bool GiNaC::integration_kernel::uses_Laurent_series () const [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented in [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

References [uses\\_Laurent\\_series\(\)](#).

Referenced by [series\\_coeff\(\)](#), and [uses\\_Laurent\\_series\(\)](#).

### 8.78.2.7 series\_coeff\_impl()

```
cln::cl_N GiNaC::integration_kernel::series_coeff_impl (
    int i) const [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented in [GiNaC::basic\\_log\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), and [GiNaC::multiple\\_polylog\\_kernel](#).

References [series\\_coeff\\_impl\(\)](#).

Referenced by [series\\_coeff\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.78.2.8 get\_cache\_size()

```
size_t GiNaC::integration_kernel::get_cache_size (
    void ) const
```

Returns the current size of the cache.

References [get\\_cache\\_size\(\)](#), and [series\\_vec](#).

Referenced by [get\\_cache\\_size\(\)](#).

### 8.78.2.9 set\_cache\_step()

```
void GiNaC::integration_kernel::set_cache_step (
    int cache_steps) const
```

Sets the step size by which the cache is increased.

References [cache\\_step\\_size](#), and [set\\_cache\\_step\(\)](#).

Referenced by [set\\_cache\\_step\(\)](#).

**8.78.2.10 get\_series\_coeff()**

```
ex GiNaC::integration_kernel::get_series_coeff (
    int i) const
```

Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.

References [GiNaC::basic::ex](#), [get\\_series\\_coeff\(\)](#), and [series\\_coeff\(\)](#).

Referenced by [get\\_series\\_coeff\(\)](#).

**8.78.2.11 series\_coeff()**

```
cln::cl_N GiNaC::integration_kernel::series_coeff (
    int i) const
```

Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.

The method `series_coeff_impl` can be used, if a single coefficient can be computed independently of the others.

The method `Laurent_series` can be used, if it is more efficient to compute a Laurent series in one shot and to determine a range of coefficients from this Laurent series.

References [cache\\_step\\_size](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [Laurent\\_series\(\)](#), [series\\_coeff\(\)](#), [series\\_coeff\\_impl\(\)](#), [series\\_vec](#), and [uses\\_Laurent\\_series\(\)](#).

Referenced by [get\\_numerical\\_value\\_impl\(\)](#), [get\\_series\\_coeff\(\)](#), [has\\_trailing\\_zero\(\)](#), [Laurent\\_series\(\)](#), and [series\\_coeff\(\)](#).

**8.78.2.12 get\_numerical\_value\_impl()**

```
ex GiNaC::integration_kernel::get_numerical_value_impl (
    const ex & lambda,
    const ex & pre,
    int shift,
    int N_trunc) const [protected]
```

The actual implementation for computing a numerical value for the integrand.

References [GiNaC::Digits](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [get\\_numerical\\_value\\_impl\(\)](#), and [series\\_coeff\(\)](#).

Referenced by [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::ELi\\_kernel::get\\_numerical\\_value\(\)](#), [get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::get\\_numerical\\_value\(\)](#), and [get\\_numerical\\_value\\_impl\(\)](#).

**8.78.2.13 do\_print()**

```
void GiNaC::integration_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [do\\_print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\(\)](#), and [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.78.3 Member Data Documentation

#### 8.78.3.1 `cache_step_size`

```
int GiNaC::integration_kernel::cache_step_size [mutable], [protected]
```

Referenced by [series\\_coeff\(\)](#), and [set\\_cache\\_step\(\)](#).

#### 8.78.3.2 `series_vec`

```
std::vector<cln::cl_N> GiNaC::integration_kernel::series_vec [mutable], [protected]
```

Referenced by [get\\_cache\\_size\(\)](#), and [series\\_coeff\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.79 GiNaC::is\_not\_a\_clifford Struct Reference

Predicate for finding non-clifford objects.

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

#### 8.79.1 Detailed Description

Predicate for finding non-clifford objects.

### 8.79.2 Member Function Documentation

#### 8.79.2.1 `operator()`

```
bool GiNaC::is_not_a_clifford::operator() (
    const ex & e) [inline]
```

References [GiNaC::is\\_a\(\)](#).

The documentation for this struct was generated from the following file:

- [clifford.cpp](#)

## 8.80 GiNaC::is\_summation\_idx Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 8.80.1 Member Function Documentation

#### 8.80.1.1 [operator\(\)](#)

```
bool GiNaC::is_summation_idx::operator() (
    const ex & e) [inline]
```

References [GiNaC::is\\_dummy\\_pair\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 8.81 GiNaC::iterated\_integral2\_SERIAL Class Reference

Complete elliptic integral of the first kind.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.81.1 Detailed Description

Complete elliptic integral of the first kind.

Complete elliptic integral of the second kind. Iterated integral.

### 8.81.2 Member Data Documentation

#### 8.81.2.1 [serial](#)

```
unsigned GiNaC::iterated_integral2_SERIAL::serial [static]
```

**Initial value:**

```
=
function::register_new(function_options("iterated_integral", 2).
    eval_func(iterated_integral2_eval).
    evalf_func(iterated_integral2_evalf).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 8.82 GiNaC::iterated\_integral3\_SERIAL Class Reference

Iterated integral with explicit truncation.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.82.1 Detailed Description

Iterated integral with explicit truncation.

### 8.82.2 Member Data Documentation

#### 8.82.2.1 serial

```
unsigned GiNaC::iterated_integral3_SERIAL::serial [static]
```

**Initial value:**

```
=
function::register_new(function_options("iterated_integral", 3).
    eval_func(iterated_integral3_eval).
    evalf_func(iterated_integral3_evalf).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

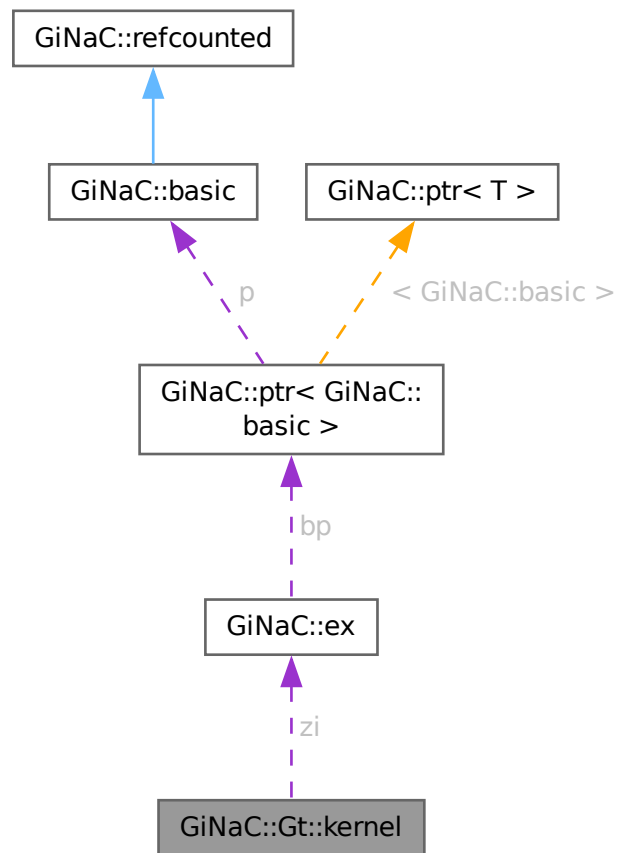
The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 8.83 GiNaC::Gt::kernel Struct Reference

```
#include <Gt.h>
```

Collaboration diagram for GiNaC::Gt::kernel:



### Public Member Functions

- `kernel ()`
- `kernel (size_t ni, ex zi, int deform=0)`
- `kernel (lst arg)`

### Public Attributes

- `size_t ni`
- `ex zi`
- `int deform`

### Private Member Functions

- `void check ()`

**Friends**

- class [Gt](#)

**8.83.1 Constructor & Destructor Documentation****8.83.1.1 `kernel()` [1/3]**

```
GiNaC::Gt::kernel::kernel ()
```

References [deform](#), [GiNaC::basic::ex](#), [ni](#), and [zi](#).

Referenced by [kernel\(\)](#).

**8.83.1.2 `kernel()` [2/3]**

```
GiNaC::Gt::kernel::kernel (
    size_t ni,
    ex zi,
    int deform = 0)
```

References [check\(\)](#), [deform](#), [GiNaC::basic::ex](#), [kernel\(\)](#), [ni](#), and [zi](#).

**8.83.1.3 `kernel()` [3/3]**

```
GiNaC::Gt::kernel::kernel (
    lst arg)
```

References [check\(\)](#), [deform](#), [ni](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), and [zi](#).

**8.83.2 Member Function Documentation****8.83.2.1 `check()`**

```
void GiNaC::Gt::kernel::check () [private]
```

References [deform](#), [ni](#), and [zi](#).

Referenced by [kernel\(\)](#), and [kernel\(\)](#).

**8.83.3 Friends And Related Symbol Documentation****8.83.3.1 `Gt`**

```
friend class Gt [friend]
```

References [Gt](#).

Referenced by [Gt](#).



### 8.83.4 Member Data Documentation

#### 8.83.4.1 ni

`size_t GiNaC::Gt::kernel::ni`

Referenced by [check\(\)](#), [kernel\(\)](#), [kernel\(\)](#), [kernel\(\)](#), and [GiNaC::Gt::tauToFundamental\(\)](#).

#### 8.83.4.2 zi

`ex GiNaC::Gt::kernel::zi`

Referenced by [check\(\)](#), [kernel\(\)](#), [kernel\(\)](#), and [kernel\(\)](#).

#### 8.83.4.3 deform

`int GiNaC::Gt::kernel::deform`

Referenced by [check\(\)](#), [kernel\(\)](#), [kernel\(\)](#), and [kernel\(\)](#).

The documentation for this struct was generated from the following files:

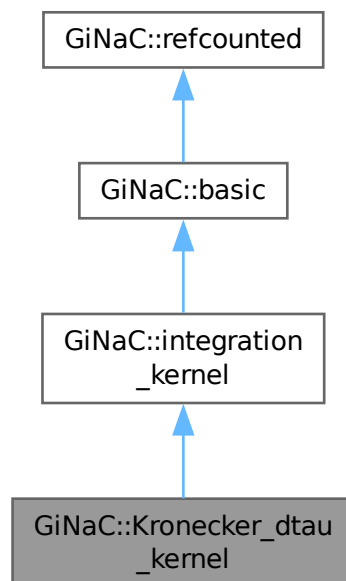
- [Gt.h](#)
- [Gt.cpp](#)

## 8.84 GiNaC::Kronecker\_dtau\_kernel Class Reference

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dtau\_kernel:





## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex z](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- [std::vector< cln::cl\\_N >](#) [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.84.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},\tau}(z_j) = \frac{C_n K(n-1)}{(2\pi i)^n} g^{(n)}(z_j, K\tau) \frac{d\bar{q}}{\bar{q}}$$

## 8.84.2 Constructor & Destructor Documentation

### 8.84.2.1 Kronecker\_dtau\_kernel()

```
GiNaC::Kronecker_dtau_kernel::Kronecker_dtau_kernel (
    const ex & n,
    const ex & z,
    const ex & K = numeric(1),
    const ex & C_norm = numeric(1))
```

References [C\\_norm](#), [K](#), [Kronecker\\_dtau\\_kernel\(\)](#), [n](#), and [z](#).

Referenced by [Kronecker\\_dtau\\_kernel\(\)](#).

## 8.84.3 Member Function Documentation

### 8.84.3.1 nops()

```
size_t GiNaC::Kronecker_dtau_kernel::nops () const \[override\], \[virtual\]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.84.3.2 op()

```
ex GiNaC::Kronecker_dtau_kernel::op (
    size_t i) const \[override\], \[virtual\]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), and [z](#).

### 8.84.3.3 let\_op()

```
ex & GiNaC::Kronecker_dtau_kernel::let_op (
    size_t i) \[override\], \[virtual\]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), and [z](#).

#### 8.84.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dtau_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [z](#).

#### 8.84.3.5 get\_numerical\_value()

```
ex GiNaC::Kronecker_dtau_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of the  $g^{(n)}(z, K \cdot \tau)$ , where  $\tau$  is given by  $qbar$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_val](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), and [z](#).

#### 8.84.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Kronecker_dtau_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::l](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [z](#).

#### 8.84.3.7 do\_print()

```
void GiNaC::Kronecker_dtau_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [C\\_norm](#), [K](#), [n](#), [GiNaC::print\\_context::s](#), and [z](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).



## 8.84.4 Member Data Documentation

### 8.84.4.1 n

`ex` GiNaC::Kronecker\_dtau\_kernel::n [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dtau\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.84.4.2 z

`ex` GiNaC::Kronecker\_dtau\_kernel::z [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dtau\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.84.4.3 K

`ex` GiNaC::Kronecker\_dtau\_kernel::K [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dtau\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.84.4.4 C\_norm

`ex` GiNaC::Kronecker\_dtau\_kernel::C\_norm [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dtau\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

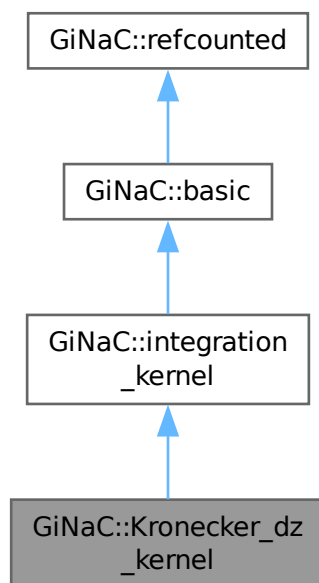
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.85 GiNaC::Kronecker\_dz\_kernel Class Reference

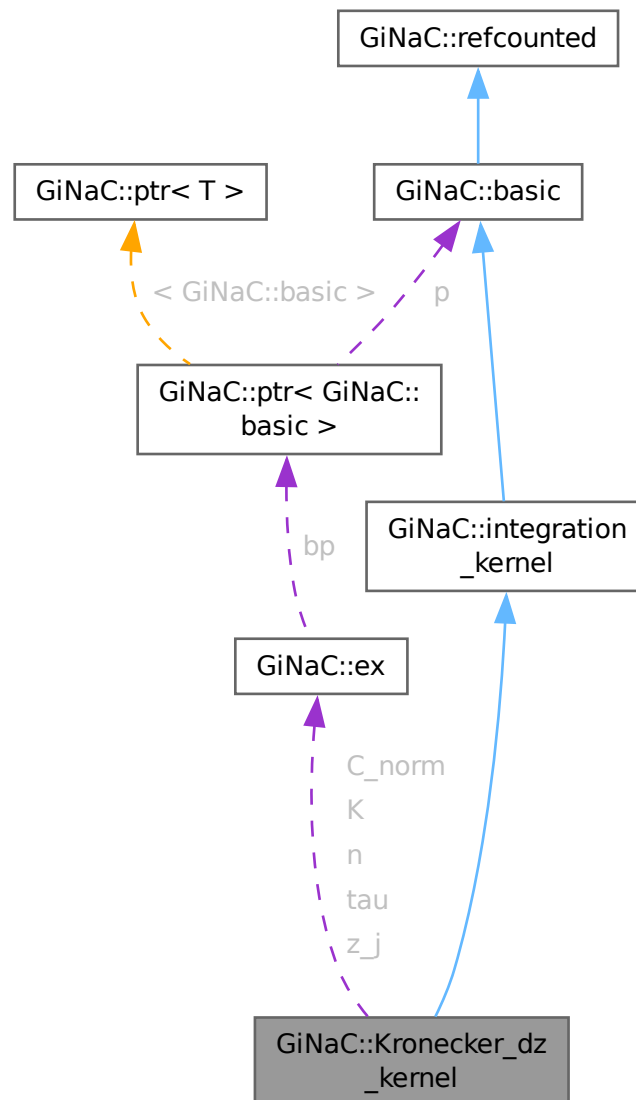
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dz\_kernel:



Collaboration diagram for GiNaC::Kronecker\_dz\_kernel:



### Public Member Functions

- `Kronecker_dz_kernel` (const `ex` &`n`, const `ex` &`z_j`, const `ex` &`tau`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- bool `is_numeric` (void) const override

*This routine returns true, if the integration kernel can be evaluated numerically.*

- `ex get_numerical_value` (const `ex` &z, int N\_trunc=0) const override

*Returns the value of the  $g^{(n-1)}(z-z_j, K\tau)$ .*

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &r, int order, unsigned options=0) const override

*Default implementation of `ex::series()`.*

- virtual bool `has_trailing_zero` (void) const

*This routine returns true, if the integration kernel has a trailing zero.*

- virtual `ex Laurent_series` (const `ex` &x, int order) const

*Returns the Laurent series, starting possibly with the pole term.*

- `size_t get_cache_size` (void) const

*Returns the current size of the cache.*

- void `set_cache_step` (int cache\_steps) const

*Sets the step size by which the cache is increased.*

- `ex get_series_coeff` (int i) const

*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*

- `cln::cl_N series_coeff` (int i) const

*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()

*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*

- `basic` (const `basic` &other)

- const `basic` & `operator=` (const `basic` &other)

*basic assignment operator: the other object might be of a derived class.*

- virtual `basic * duplicate` () const

*Create a clone of this object on the heap.*

- virtual `ex eval` () const

*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf` () const

*Evaluate object numerically.*

- virtual `ex evalm` () const

*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ` () const

*Evaluate integrals, if result is known.*

- virtual `ex eval_indexed` (const `basic` &i) const

*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

- virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*

- virtual void `dbgprint` () const

*Little wrapper around `print` to be called within a debugger.*

- virtual void `dbgprinttree` () const

*Little wrapper around `printtree` to be called within a debugger.*

- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*

- virtual bool `info` (unsigned inf) const

*Information about the object.*

- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const

*Test for occurrence of a pattern.*

- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*

- virtual `ex subs` (const `exmap` &m, unsigned options=0) const

*Substitute a set of objects by arbitrary expressions.*

- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*

- virtual `ex expand` (unsigned options=0) const

*Expand expression, i.e.*

- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*

- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const

- template<class T>

void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex z\\_j](#)
- [ex tau](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< cln::cl\_N > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.85.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},z}(z_j, \tau) = C_n (2\pi i)^{2-n} g^{(n-1)}(z - z_j, K\tau) dz$$

## 8.85.2 Constructor & Destructor Documentation

### 8.85.2.1 Kronecker\_dz\_kernel()

```
GiNaC::Kronecker_dz_kernel::Kronecker_dz_kernel (
    const ex & n,
    const ex & z_j,
    const ex & tau,
    const ex & K = numeric(1),
    const ex & C_norm = numeric(1))
```

References [C\\_norm](#), [K](#), [Kronecker\\_dz\\_kernel\(\)](#), [n](#), [tau](#), and [z\\_j](#).

Referenced by [Kronecker\\_dz\\_kernel\(\)](#).

## 8.85.3 Member Function Documentation

### 8.85.3.1 nops()

```
size_t GiNaC::Kronecker_dz_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.85.3.2 op()

```
ex GiNaC::Kronecker_dz_kernel::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), [tau](#), and [z\\_j](#).

### 8.85.3.3 let\_op()

```
ex & GiNaC::Kronecker_dz_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), [tau](#), and [z\\_j](#).



### 8.85.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dz_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [tau](#), and [z\\_j](#).

### 8.85.3.5 get\_numerical\_value()

```
ex GiNaC::Kronecker_dz_kernel::get_numerical_value (
    const ex & z,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of the  $g^{(n-1)}(z-z_j, K*\tau)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#), [n](#), [GiNaC::Pi](#), and [GiNaC::pow\(\)](#).

### 8.85.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Kronecker_dz_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::l](#), [GiNaC::is\\_even\(\)](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [tau](#), and [z\\_j](#).

### 8.85.3.7 do\_print()

```
void GiNaC::Kronecker_dz_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [C\\_norm](#), [K](#), [n](#), [GiNaC::print\\_context::s](#), [tau](#), and [z\\_j](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.85.4 Member Data Documentation

### 8.85.4.1 `n`

`ex GiNaC::Kronecker_dz_kernel::n` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dz\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.85.4.2 `z_j`

`ex GiNaC::Kronecker_dz_kernel::z_j` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dz\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.85.4.3 `tau`

`ex GiNaC::Kronecker_dz_kernel::tau` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dz\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.85.4.4 `K`

`ex GiNaC::Kronecker_dz_kernel::K` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dz\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 8.85.4.5 `C_norm`

`ex GiNaC::Kronecker_dz_kernel::C_norm` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Kronecker\\_dz\\_kernel\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.86 GiNaC::lanczos\_coeffs Class Reference

### Public Member Functions

- [lanczos\\_coeffs](#) ()
- bool [sufficiently\\_accurate](#) (int digits)
- int [get\\_order](#) () const
- cln::cl\_N [calc\\_lanczos\\_A](#) (const cln::cl\_N &) const

### Private Attributes

- `std::vector< cln::cl_N > * current\_vector`

### Static Private Attributes

- `static std::vector< cln::cl_N > * coeffs = nullptr`

## 8.86.1 Constructor & Destructor Documentation

### 8.86.1.1 `lanczos_coeffs()`

`GiNaC::lanczos_coeffs::lanczos_coeffs ()`

References [coeffs](#).

## 8.86.2 Member Function Documentation

### 8.86.2.1 `sufficiently_accurate()`

`bool GiNaC::lanczos_coeffs::sufficiently_accurate (int digits)`

References [coeffs](#), and [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 8.86.2.2 `get_order()`

`int GiNaC::lanczos_coeffs::get_order () const [inline]`

References [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 8.86.2.3 `calc_lanczos_A()`

`cln::cl_N GiNaC::lanczos_coeffs::calc_lanczos_A (const cln::cl_N & x) const`

References [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 8.86.3 Member Data Documentation

#### 8.86.3.1 coeffs

```
std::vector< cln::cl_N > * GiNaC::lanczos_coeffs::coeffs = nullptr [static], [private]
```

Referenced by [lanczos\\_coeffs\(\)](#), and [sufficiently\\_accurate\(\)](#).

#### 8.86.3.2 current\_vector

```
std::vector<cln::cl_N>* GiNaC::lanczos_coeffs::current_vector [private]
```

Referenced by [calc\\_lanczos\\_A\(\)](#), [get\\_order\(\)](#), and [sufficiently\\_accurate\(\)](#).

The documentation for this class was generated from the following file:

- [numeric.cpp](#)

## 8.87 std::less< GiNaC::ptr< T > > Struct Template Reference

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

```
#include <ptr.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [GiNaC::ptr< T >](#) &lhs, const [GiNaC::ptr< T >](#) &rhs) const

#### 8.87.1 Detailed Description

```
template<class T>
struct std::less< GiNaC::ptr< T > >
```

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

for the use as std::map keys).

### 8.87.2 Member Function Documentation

#### 8.87.2.1 operator>()

```
template<class T>
bool std::less< GiNaC::ptr< T > >::operator() (
    const GiNaC::ptr< T > & lhs,
    const GiNaC::ptr< T > & rhs) const [inline]
```

The documentation for this struct was generated from the following file:

- [ptr.h](#)

## 8.88 GiNaC::library\_init Class Reference

Helper class to initialize the library.

```
#include <ex.h>
```

### Public Member Functions

- [library\\_init](#) ()  
*Ctor of static initialization helpers.*
- [~library\\_init](#) ()  
*Dtor of static initialization helpers.*

### Static Private Member Functions

- static void [init\\_unarchivers](#) ()

### Static Private Attributes

- static int [count](#) = 0  
*How many static objects were created?*

### 8.88.1 Detailed Description

Helper class to initialize the library.

There must be one static object of this class in every object file that makes use of our flyweights in order to guarantee proper initialization. Hence we put it into this file which is included by every relevant file anyways. This is modeled after section [ios::Init] of the C++ standard, where cout and friends are set up.

See also

[utils.cpp](#)

## 8.88.2 Constructor & Destructor Documentation

### 8.88.2.1 library\_init()

```
GiNaC::library_init::library_init ()
```

Ctor of static initialization helpers.

The fist call to this is going to initialize the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), [GiNaC::\\_num0\\_bp](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num120\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num15\\_p](#), [GiNaC::\\_num18\\_p](#), [GiNaC::\\_num1\\_2\\_p](#), [GiNaC::\\_num1\\_3\\_p](#), [GiNaC::\\_num1\\_4\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num20\\_p](#), [GiNaC::\\_num24\\_p](#), [GiNaC::\\_num25\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num30\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num48\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num60\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_120\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_15\\_p](#), [GiNaC::\\_num\\_18\\_p](#), [GiNaC::\\_num\\_1\\_2\\_p](#), [GiNaC::\\_num\\_1\\_3\\_p](#), [GiNaC::\\_num\\_1\\_4\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_20\\_p](#), [GiNaC::\\_num\\_24\\_p](#), [GiNaC::\\_num\\_25\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_30\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_48\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_60\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), [GiNaC::\\_num\\_9\\_p](#), [count](#), and [GiNaC::dynamallocate\(\)](#).

### 8.88.2.2 ~library\_init()

```
GiNaC::library_init::~~library_init ()
```

Dtor of static initialization helpers.

The last call to this is going to shut down the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), and [count](#).

## 8.88.3 Member Function Documentation

### 8.88.3.1 init\_unarchivers()

```
void GiNaC::library_init::init_unarchivers () [static], [private]
```

## 8.88.4 Member Data Documentation

### 8.88.4.1 count

```
int GiNaC::library_init::count = 0 [static], [private]
```

How many static objects were created?

Only the first one must create the static flyweights on the heap.

Referenced by [library\\_init\(\)](#), and [~library\\_init\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [utils.cpp](#)

## 8.89 GiNaC::make\_flat\_inserter Class Reference

Class to handle the renaming of dummy indices.

```
#include <expairseq.h>
```

### Public Member Functions

- [make\\_flat\\_inserter](#) (const [epvector](#) &epv, bool b)
- [make\\_flat\\_inserter](#) (const [exvector](#) &v, bool b)
- [ex handle\\_factor](#) (const [ex](#) &x, const [ex](#) &coeff)

### Private Member Functions

- void [combine\\_indices](#) (const [exvector](#) &dummies\_of\_factor)

### Private Attributes

- bool [do\\_renaming](#)
- [exvector](#) [used\\_indices](#)

### 8.89.1 Detailed Description

Class to handle the renaming of dummy indices.

It holds a vector of indices that are being used in the expression so far. If the same index occurs again as a dummy index in a factor, it is to be renamed. Unless dummy index renaming was switched off, of course ;-)

## 8.89.2 Constructor & Destructor Documentation

### 8.89.2.1 `make_flat_inserter()` [1/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const epvector & epv,
    bool b) [inline]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [combine\\_indices\(\)](#), and [do\\_renaming](#).

### 8.89.2.2 `make_flat_inserter()` [2/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const exvector & v,
    bool b) [inline]
```

References [combine\\_indices\(\)](#), and [do\\_renaming](#).

## 8.89.3 Member Function Documentation

### 8.89.3.1 `handle_factor()`

```
ex GiNaC::make_flat_inserter::handle_factor (
    const ex & x,
    const ex & coeff) [inline]
```

References [GiNaC::coeff\(\)](#), [combine\\_indices\(\)](#), [do\\_renaming](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), and [used\\_indices](#).

Referenced by [GiNaC::ncmul::eval\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), and [GiNaC::expairseq::make\\_flat\(\)](#).

### 8.89.3.2 `combine_indices()`

```
void GiNaC::make_flat_inserter::combine_indices (
    const exvector & dummies_of_factor) [inline], [private]
```

References [used\\_indices](#).

Referenced by [handle\\_factor\(\)](#), [make\\_flat\\_inserter\(\)](#), and [make\\_flat\\_inserter\(\)](#).

## 8.89.4 Member Data Documentation

### 8.89.4.1 `do_renaming`

```
bool GiNaC::make_flat_inserter::do_renaming [private]
```

Referenced by [handle\\_factor\(\)](#), [make\\_flat\\_inserter\(\)](#), and [make\\_flat\\_inserter\(\)](#).



#### 8.89.4.2 used\_indices

`exvector` GiNaC::make\_flat\_inserter::used\_indices [private]

Referenced by [combine\\_indices\(\)](#), and [handle\\_factor\(\)](#).

The documentation for this class was generated from the following file:

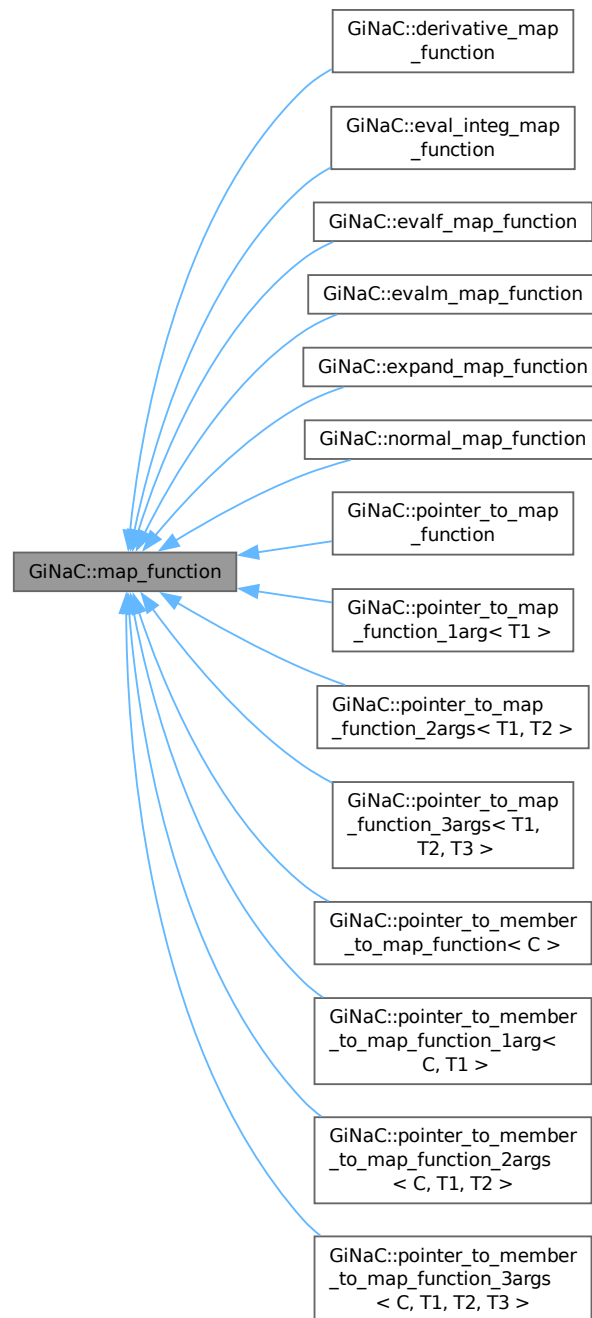
- [expairseq.h](#)

## 8.90 GiNaC::map\_function Struct Reference

Function object for `map()`.

```
#include <basic.h>
```

Inheritance diagram for `GiNaC::map_function`:



### Public Types

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

### Public Member Functions

- virtual `~map_function` ()
- virtual `ex operator()` (const `ex` &`e`)=0

### 8.90.1 Detailed Description

Function object for map().

### 8.90.2 Member Typedef Documentation

#### 8.90.2.1 argument\_type

```
typedef const ex& GiNaC::map\_function::argument\_type
```

#### 8.90.2.2 result\_type

```
typedef ex GiNaC::map\_function::result\_type
```

### 8.90.3 Constructor & Destructor Documentation

#### 8.90.3.1 ~map\_function()

```
virtual GiNaC::map\_function::~~map\_function () \[inline\], \[virtual\]
```

### 8.90.4 Member Function Documentation

#### 8.90.4.1 operator>()

```
virtual ex GiNaC::map\_function::operator() (  
    const ex & e) \[pure virtual\]
```

Implemented in [GiNaC::derivative\\_map\\_function](#), [GiNaC::eval\\_integ\\_map\\_function](#), [GiNaC::evalf\\_map\\_function](#), [GiNaC::evalm\\_map\\_function](#), [GiNaC::expand\\_map\\_function](#), [GiNaC::normal\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#).

The documentation for this struct was generated from the following file:

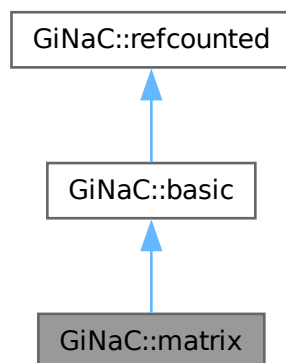
- [basic.h](#)

## 8.91 GiNaC::matrix Class Reference

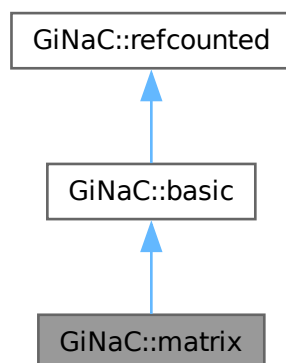
Symbolic matrices.

```
#include <matrix.h>
```

Inheritance diagram for GiNaC::matrix:



Collaboration diagram for GiNaC::matrix:



### Public Member Functions

- [matrix](#) (unsigned r, unsigned c)  
*Very common ctor.*
- [matrix](#) (unsigned r, unsigned c, const [lst](#) &l)

- Construct matrix from (flat) list of elements.*

  - `matrix` (`std::initializer_list< std::initializer_list< ex > > l`)

*Construct a matrix from an 2 dimensional initializer list.*
- `size_t nops` () const override

*nops is defined to be rows x columns.*
- `ex op` (`size_t i`) const override

*returns matrix entry at position (i/col, icol).*
- `ex & let_op` (`size_t i`) override

*returns writable matrix entry at position (i/col, icol).*
- `ex evalm` () const override

*Evaluate sums, products and integer powers of matrices.*
- `ex subs` (const `exmap` &`m`, unsigned `options=0`) const override

*Substitute a set of objects by arbitrary expressions.*
- `ex eval_indexed` (const `basic` &`i`) const override

*Automatic symbolic evaluation of an indexed matrix.*
- `ex add_indexed` (const `ex` &`self`, const `ex` &`other`) const override

*Sum of two indexed matrices.*
- `ex scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const override

*Product of an indexed matrix with a number.*
- `bool contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector` &`v`) const override

*Contraction of an indexed matrix with something else.*
- `ex conjugate` () const override

*Complex conjugate every matrix entry.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `void archive` (`archive_node` &`n`) const override

*Save (a.k.a.*
- `void read_archive` (const `archive_node` &`n`, `lst` &`syms`) override

*Read (a.k.a.*
- unsigned `rows` () const

*Get number of rows.*
- unsigned `cols` () const

*Get number of columns.*
- `matrix add` (const `matrix` &`other`) const

*Sum of matrices.*
- `matrix sub` (const `matrix` &`other`) const

*Difference of matrices.*
- `matrix mul` (const `matrix` &`other`) const

*Product of matrices.*
- `matrix mul` (const `numeric` &`other`) const

*Product of matrix and scalar.*
- `matrix mul_scalar` (const `ex` &`other`) const

*Product of matrix and scalar expression.*
- `matrix pow` (const `ex` &`expn`) const

*Power of a matrix.*
- const `ex & operator()` (unsigned `ro`, unsigned `co`) const

*operator() to access elements for reading.*
- `ex & operator()` (unsigned `ro`, unsigned `co`)

*operator() to access elements for writing.*
- `matrix & set` (unsigned `ro`, unsigned `co`, const `ex` &`value`)
- `matrix transpose` () const

- Transposed of an  $m \times n$  matrix, producing a new  $n \times m$  matrix object that represents the transposed.*

  - `ex determinant` (unsigned algo=`determinant_algo::automatic`) const

*Determinant of square matrix.*
- `ex trace` () const

*Trace of a matrix.*
- `ex charpoly` (const `ex` &lambda) const

*Characteristic Polynomial.*
- `matrix inverse` () const

*Inverse of this matrix, with automatic algorithm selection.*
- `matrix inverse` (unsigned algo) const

*Inverse of this matrix.*
- `matrix solve` (const `matrix` &vars, const `matrix` &rhs, unsigned algo=`solve_algo::automatic`) const

*Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.*
- unsigned `rank` () const

*Compute the rank of this matrix.*
- unsigned `rank` (unsigned `solve_algo`) const

*Compute the rank of this matrix using the given algorithm, which should be a member of enum `solve_algo`.*
- bool `is_zero_matrix` () const

*Function to check that all elements of the matrix are zero.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()

*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)

*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const

*Create a clone of this object on the heap.*
- virtual `ex eval` () const

*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const

*Evaluate object numerically.*
- virtual `ex eval_integ` () const

*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const

*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const

- Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `return_type_t return_type_tinfo` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const
  - Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [matrix](#) (unsigned r, unsigned c, const [exvector](#) &m2)  
*Ctor from representation, for internal use only.*
- [matrix](#) (unsigned r, unsigned c, [exvector](#) &&m2)
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [ex\\_determinant\\_minor](#) () const  
*Recursive determinant for small matrices having at least one symbolic entry.*
- std::vector< unsigned > [echelon\\_form](#) (unsigned algo, int n)
- int [gauss\\_elimination](#) (const bool det=false)  
*Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.*
- int [division\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.*
- int [fraction\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.*
- std::vector< unsigned > [markowitz\\_elimination](#) (unsigned n)
- int [pivot](#) (unsigned ro, unsigned co, bool symbolic=true)  
*Partial pivoting method for matrix elimination schemes.*
- void [print\\_elements](#) (const [print\\_context](#) &c, const char \*row\_start, const char \*row\_end, const char \*row↵\_sep, const char \*col\_sep) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex\\_eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*



## Protected Attributes

- unsigned [row](#)  
*number of rows*
- unsigned [col](#)  
*number of columns*
- [exvector](#) [m](#)  
*representation (cols indexed first)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.91.1 Detailed Description

Symbolic matrices.

### 8.91.2 Constructor & Destructor Documentation

#### 8.91.2.1 [matrix\(\)](#) [1/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c)
```

Very common ctor.

Initializes to r x c-dimensional zero-matrix.

#### Parameters

<i>r</i>	number of rows
<i>c</i>	number of cols

References [GiNaC::\\_ex0](#), [col](#), [m](#), [matrix\(\)](#), [GiNaC::status\\_flags::not\\_shareable](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [fraction\\_free\\_elimination\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [inverse\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [pow\(\)](#), [rank\(\)](#), [real\\_part\(\)](#), [scalar\\_mul\\_indexed\(\)](#), [set\(\)](#), [solve\(\)](#), [sub\(\)](#), [subs\(\)](#), and [transpose\(\)](#).

**8.91.2.2 matrix() [2/5]**

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const lst & l)
```

Construct matrix from (flat) list of elements.

If the list has fewer elements than the matrix, the remaining matrix elements are set to zero. If the list has more elements than the matrix, the excessive elements are thrown away.

References [GiNaC::\\_ex0](#), [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

**8.91.2.3 matrix() [3/5]**

```
GiNaC::matrix::matrix (
    std::initializer_list< std::initializer_list< ex > > l)
```

Construct a matrix from an 2 dimensional initializer list.

Throws an exception if some row has a different length than all the others.

References [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

**8.91.2.4 matrix() [4/5]**

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const exvector & m2) [protected]
```

Ctor from representation, for internal use only.

References [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

**8.91.2.5 matrix() [5/5]**

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    exvector && m2) [protected]
```

References [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

## 8.91.3 Member Function Documentation

### 8.91.3.1 nops()

```
size_t GiNaC::matrix::nops () const [override], [virtual]
```

nops is defined to be rows x columns.

Reimplemented from [GiNaC::basic](#).

References [col](#), and [row](#).

Referenced by [let\\_op\(\)](#), [GiNaC::nops\(\)](#), and [op\(\)](#).

### 8.91.3.2 op()

```
ex GiNaC::matrix::op (
    size_t i) const [override], [virtual]
```

returns matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

Referenced by [contract\\_with\(\)](#).

### 8.91.3.3 let\_op()

```
ex & GiNaC::matrix::let_op (
    size_t i) [override], [virtual]
```

returns writable matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

### 8.91.3.4 evalm()

```
ex GiNaC::matrix::evalm () const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.91.3.5 subs()

```
ex GiNaC::matrix::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), [row](#), and [subs\(\)](#).

Referenced by [fraction\\_free\\_elimination\(\)](#), and [subs\(\)](#).

### 8.91.3.6 eval\_indexed()

```
ex GiNaC::matrix::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed matrix.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [col](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [row](#), and [trace\(\)](#).

### 8.91.3.7 add\_indexed()

```
ex GiNaC::matrix::add_indexed (
    const ex & self,
    const ex & other) const [override], [virtual]
```

Sum of two indexed matrices.

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [col](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [matrix\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [row](#), and [transpose\(\)](#).

### 8.91.3.8 scalar\_mul\_indexed()

```
ex GiNaC::matrix::scalar_mul_indexed (
    const ex & self,
    const numeric & other) const [override], [virtual]
```

Product of an indexed matrix with a number.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [matrix\(\)](#), [mul\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

### 8.91.3.9 contract\_with()

```
bool GiNaC::matrix::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed matrix with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [col](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [matrix\(\)](#), [mul\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [row](#), and [transpose\(\)](#).

### 8.91.3.10 conjugate()

```
ex GiNaC::matrix::conjugate () const [override], [virtual]
```

Complex conjugate every matrix entry.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [col](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), and [row](#).

### 8.91.3.11 real\_part()

```
ex GiNaC::matrix::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), and [row](#).

### 8.91.3.12 imag\_part()

```
ex GiNaC::matrix::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), and [row](#).

### 8.91.3.13 archive()

```
void GiNaC::matrix::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [col](#), [m](#), and [row](#).

### 8.91.3.14 read\_archive()

```
void GiNaC::matrix::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::basic::ex](#), [GiNaC::archive\\_node::find\\_ex\\_by\\_loc\(\)](#), [GiNaC::archive\\_node::find\\_property\\_range\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [m](#), and [row](#).

### 8.91.3.15 match\_same\_type()

```
bool GiNaC::matrix::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [col](#), [cols\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [matrix\(\)](#), [row](#), and [rows\(\)](#).

### 8.91.3.16 return\_type()

```
unsigned GiNaC::matrix::return_type() const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

### 8.91.3.17 rows()

```
unsigned GiNaC::matrix::rows () const [inline]
```

Get number of rows.

References [row](#).

Referenced by [GiNaC::clifford\\_unit\(\)](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::rows\(\)](#), [solve\(\)](#), [GiNaC::sub\\_matrix\(\)](#), and [transpose\(\)](#).

**8.91.3.18 cols()**

```
unsigned GiNaC::matrix::cols () const [inline]
```

Get number of columns.

References [col](#).

Referenced by [GiNaC::clifford\\_unit\(\)](#), [GiNaC::cols\(\)](#), [determinant\\_minor\(\)](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [solve\(\)](#), [GiNaC::sub\\_matrix\(\)](#), and [transpose\(\)](#).

**8.91.3.19 add()**

```
matrix GiNaC::matrix::add (
    const matrix & other) const
```

Sum of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

Referenced by [add\\_indexed\(\)](#), and [GiNaC::add::evalm\(\)](#).

**8.91.3.20 sub()**

```
matrix GiNaC::matrix::sub (
    const matrix & other) const
```

Difference of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

**8.91.3.21 mul() [1/2]**

```
matrix GiNaC::matrix::mul (
    const matrix & other) const
```

Product of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [col](#), [cols\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [matrix\(\)](#), [row](#), and [rows\(\)](#).

Referenced by [charpoly\(\)](#), [contract\\_with\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [pow\(\)](#), and [scalar\\_mul\\_indexed\(\)](#).

**8.91.3.22 mul()** [2/2]

```
matrix GiNaC::matrix::mul (
    const numeric & other) const
```

Product of matrix and scalar.

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

**8.91.3.23 mul\_scalar()**

```
matrix GiNaC::matrix::mul_scalar (
    const ex & other) const
```

Product of matrix and scalar expression.

References [col](#), [GiNaC::return\\_types::commutative](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), [GiNaC::ex::return\\_type\(\)](#), and [row](#).

Referenced by [GiNaC::mul::evalm\(\)](#).

**8.91.3.24 pow()**

```
matrix GiNaC::matrix::pow (
    const ex & expn) const
```

Power of a matrix.

Currently handles integer exponents only.

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [col](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [inverse\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [matrix\(\)](#), [mul\(\)](#), [GiNaC::info\\_flags::negative](#), and [row](#).

**8.91.3.25 operator>()** [1/2]

```
const ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co) const
```

[operator\(\)](#) to access elements for reading.

**Parameters**

<i>ro</i>	row of element
<i>co</i>	column of element

**Exceptions**

<i>range_error</i>	(index out of range)
--------------------	----------------------

References [col](#), [GiNaC::basic::ex](#), [m](#), and [row](#).



### 8.91.3.26 operator>() [2/2]

```
ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co)
```

operator() to access elements for writing.

#### Parameters

<i>ro</i>	row of element
-----------	----------------

<i>co</i>	column of element
-----------	-------------------

### Exceptions

<i>range_error</i>	(index out of range)
--------------------	----------------------

References [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [m](#), and [row](#).

#### 8.91.3.27 set()

```
matrix & GiNaC::matrix::set (
    unsigned ro,
    unsigned co,
    const ex & value) [inline]
```

References [GiNaC::basic::ex](#), and [matrix\(\)](#).

#### 8.91.3.28 transpose()

```
matrix GiNaC::matrix::transpose () const
```

Transposed of an m x n matrix, producing a new n x m matrix object that represents the transposed.

References [cols\(\)](#), [m](#), [matrix\(\)](#), and [rows\(\)](#).

Referenced by [add\\_indexed\(\)](#), [contract\\_with\(\)](#), and [GiNaC::transpose\(\)](#).

#### 8.91.3.29 determinant()

```
ex GiNaC::matrix::determinant (
    unsigned algo = determinant_algo::automatic) const
```

Determinant of square matrix.

This routine doesn't actually calculate the determinant, it only implements some heuristics about which algorithm to run. If all the elements of the matrix are elements of an integral domain the determinant is also in that integral domain and the result is expanded only. If one or more elements are from a quotient field the determinant is usually also in that quotient field and the result is normalized before it is returned. This implies that the determinant of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b), 1 \\ b/(a-b), 1 \end{bmatrix}$  is returned as unity. (In this respect, it behaves like MapleV and unlike Mathematica.)

### Parameters

<i>algo</i>	allows to chose an algorithm
-------------	------------------------------

**Returns**

the determinant as a new expression

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

**See also**

[determinant\\_algo](#)

References [GiNaC::\\_ex0](#), [GiNaC::determinant\\_algo::automatic](#), [GiNaC::determinant\\_algo::bareiss](#), [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [determinant\\_minor\(\)](#), [GiNaC::determinant\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [GiNaC::basic::ex](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::determinant\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::determinant\\_algo::laplace](#), [m](#), [matrix\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::permutation\\_sign\(\)](#), [GiNaC::info\\_flags::rational\\_function](#), and [row](#).

Referenced by [charpoly\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [GiNaC::determinant\(\)](#), and [GiNaC::resultant\(\)](#).

**8.91.3.30 trace()**

```
ex GiNaC::matrix::trace () const
```

Trace of a matrix.

The result is normalized if it is in some quotient field and expanded only otherwise. This implies that the trace of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b), x \\ y, b/(b-a) \end{bmatrix}$  is recognized to be unity.

**Returns**

the sum of diagonal elements

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

References [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::normal\(\)](#), [GiNaC::info\\_flags::rational\\_function](#), and [row](#).

Referenced by [charpoly\(\)](#), [eval\\_indexed\(\)](#), and [GiNaC::trace\(\)](#).

### 8.91.3.31 charpoly()

```
ex GiNaC::matrix::charpoly (
    const ex & lambda) const
```

Characteristic Polynomial.

Following mathematica notation the characteristic polynomial of a matrix  $M$  is defined as the determinant of  $(M - \lambda * 1)$  where  $1$  stands for the unit matrix of the same dimension as  $M$ . Note that some CASs define it with a sign inside the determinant which gives rise to an overall sign if the dimension is odd. This method returns the characteristic polynomial collected in powers of  $\lambda$  as a new expression.

#### Returns

characteristic polynomial as new expression

#### Exceptions

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

#### See also

[matrix::determinant\(\)](#)

References [col](#), [GiNaC::ex::collect\(\)](#), [determinant\(\)](#), [GiNaC::basic::ex](#), [m](#), [matrix\(\)](#), [mul\(\)](#), [GiNaC::info\\_flags::numeric](#), [row](#), and [trace\(\)](#).

Referenced by [GiNaC::charpoly\(\)](#).

### 8.91.3.32 inverse() [1/2]

```
matrix GiNaC::matrix::inverse () const
```

Inverse of this matrix, with automatic algorithm selection.

References [GiNaC::solve\\_algo::automatic](#), [inverse\(\)](#), and [matrix\(\)](#).

Referenced by [GiNaC::inverse\(\)](#), [GiNaC::inverse\(\)](#), [inverse\(\)](#), and [pow\(\)](#).

### 8.91.3.33 inverse() [2/2]

```
matrix GiNaC::matrix::inverse (
    unsigned algo) const
```

Inverse of this matrix.

#### Parameters

<i>algo</i>	selects the algorithm (one of <a href="#">solve_algo</a> )
-------------	--

#### Returns

the inverted matrix

#### Exceptions

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

<i>runtime_error</i>	(singular matrix)
----------------------	-------------------

References [GiNaC::\\_ex1](#), [col](#), [matrix\(\)](#), [row](#), and [solve\(\)](#).

### 8.91.3.34 solve()

```
matrix GiNaC::matrix::solve (
    const matrix & vars,
    const matrix & rhs,
    unsigned algo = solve_algo::automatic) const
```

Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.

#### Parameters

<i>vars</i>	$n \times p$ matrix, all elements must be symbols
<i>rhs</i>	$m \times p$ matrix
<i>algo</i>	selects the solving algorithm

#### Returns

$n \times p$  solution matrix

#### Exceptions

<i>logic_error</i>	(incompatible matrices)
<i>invalid_argument</i>	(1st argument must be matrix of symbols)
<i>runtime_error</i>	(inconsistent linear system)

#### See also

[solve\\_algo](#)

References [cols\(\)](#), [echelon\\_form\(\)](#), [GiNaC::basic::ex](#), [GiNaC::basic::info\(\)](#), [m](#), [matrix\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::rhs\(\)](#), [rows\(\)](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [inverse\(\)](#), [GiNaC::lsolve\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

### 8.91.3.35 rank() [1/2]

```
unsigned GiNaC::matrix::rank () const
```

Compute the rank of this matrix.

References [GiNaC::solve\\_algo::automatic](#), and [rank\(\)](#).

Referenced by [rank\(\)](#), [GiNaC::rank\(\)](#), and [GiNaC::rank\(\)](#).

**8.91.3.36 rank()** [2/2]

```
unsigned GiNaC::matrix::rank (
    unsigned solve_algo) const
```

Compute the rank of this matrix using the given algorithm, which should be a member of enum [solve\\_algo](#).

References [col](#), [echelon\\_form\(\)](#), [GINAC\\_ASSERT](#), [m](#), [matrix\(\)](#), and [row](#).

**8.91.3.37 is\_zero\_matrix()**

```
bool GiNaC::matrix::is_zero_matrix () const
```

Function to check that all elements of the matrix are zero.

References [m](#).

**8.91.3.38 determinant\_minor()**

```
ex GiNaC::matrix::determinant_minor () const [protected]
```

Recursive determinant for small matrices having at least one symbolic entry.

The basic algorithm, known as Laplace-expansion, is enhanced by some bookkeeping to avoid calculation of the same submatrices ("minors") more than once. According to W.M.Gentleman and S.C.Johnson this algorithm is better than elimination schemes for matrices of sparse multivariate polynomials and also for matrices of dense univariate polynomials if the matrix' dimension is larger than 7.

**Returns**

the determinant as a new expression (in expanded form)

**See also**

[matrix::determinant\(\)](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [cols\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), and [m](#).

Referenced by [determinant\(\)](#).

**8.91.3.39 echelon\_form()**

```
std::vector< unsigned > GiNaC::matrix::echelon_form (
    unsigned algo,
    int n) [protected]
```

References [GiNaC::solve\\_algo::automatic](#), [GiNaC::solve\\_algo::bareiss](#), [col](#), [GiNaC::solve\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::solve\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [m](#), [GiNaC::solve\\_algo::markowitz](#), [markowitz\\_elimination\(\)](#), [GiNaC::info\\_flags::numeric](#), and [row](#).

Referenced by [rank\(\)](#), and [solve\(\)](#).

**8.91.3.40 gauss\_elimination()**

```
int GiNaC::matrix::gauss_elimination (
    const bool det = false) [protected]
```

Perform the steps of an ordinary Gaussian elimination to bring the  $m \times n$  matrix into an upper echelon form.

The algorithm is ok for matrices with numeric coefficients but quite unsuited for symbolic matrices.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_zero\(\)](#), [m](#), [GiNaC::basic::normal\(\)](#), [pivot\(\)](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**8.91.3.41 division\_free\_elimination()**

```
int GiNaC::matrix::division_free_elimination (
    const bool det = false) [protected]
```

Perform the steps of division free elimination to bring the  $m \times n$  matrix into an upper echelon form.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [m](#), [pivot\(\)](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

### 8.91.3.42 fraction\_free\_elimination()

```
int GiNaC::matrix::fraction_free_elimination (
    const bool det = false) [protected]
```

Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.

Fraction free elimination means that divide is used straightforwardly, without computing GCDs first. This is possible, since we know the divisor at each step.

#### Parameters

<i>det</i>	may be set to true to save a lot of space if one is only interested in the last element (i.e. for calculating determinants). The others are set to zero in this case.
------------	---

#### Returns

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [cols\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::ex](#), [GiNaC::basic::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_zero\(\)](#), [m](#), [matrix\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [rows\(\)](#), and [subs\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

### 8.91.3.43 markowitz\_elimination()

```
std::vector< unsigned > GiNaC::matrix::markowitz_elimination (
    unsigned n) [protected]
```

References [GiNaC::\\_ex0](#), [col](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [GiNaC::basic::normal\(\)](#), [row](#), and [std::swap\(\)](#).

Referenced by [echelon\\_form\(\)](#).

### 8.91.3.44 pivot()

```
int GiNaC::matrix::pivot (
    unsigned ro,
    unsigned co,
    bool symbolic = true) [protected]
```

Partial pivoting method for matrix elimination schemes.

Usual pivoting (symbolic==false) returns the index to the element with the largest absolute value in column ro and swaps the current row with the one where the element was found. With (symbolic==true) it does the same thing with the first non-zero element.

#### Parameters

<i>ro</i>	is the row from where to begin
-----------	--------------------------------



<i>co</i>	is the column to be inspected
<i>symbolic</i>	signal if we want the first non-zero element to be pivoted (true) or the one with the largest absolute value (false).

**Returns**

0 if no interchange occurred, -1 if all are zero (usually signaling a degeneracy) and positive integer k means that rows ro and k were swapped.

References [GiNaC::abs\(\)](#), [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::basic::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [m](#), [row](#), and [GiNaC::swap\(\)](#).

Referenced by [division\\_free\\_elimination\(\)](#), and [gauss\\_elimination\(\)](#).

**8.91.3.45 print\_elements()**

```
void GiNaC::matrix::print_elements (
    const print\_context & c,
    const char * row_start,
    const char * row_end,
    const char * row_sep,
    const char * col_sep) const [protected]
```

References [col](#), [m](#), [row](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

**8.91.3.46 do\_print()**

```
void GiNaC::matrix::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [print\\_elements\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.91.3.47 do\_print\_latex()**

```
void GiNaC::matrix::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [col](#), [print\\_elements\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.91.3.48 do\_print\_python\_repr()

```
void GiNaC::matrix::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [print\\_elements\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.91.4 Member Data Documentation

### 8.91.4.1 row

```
unsigned GiNaC::matrix::row [protected]
```

number of rows

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)](#), [operator\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [rows\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

### 8.91.4.2 col

```
unsigned GiNaC::matrix::col [protected]
```

number of columns

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [cols\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [do\\_print\\_latex\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)](#), [operator\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

### 8.91.4.3 m

```
exvector GiNaC::matrix::m [protected]
```

representation (cols indexed first)

Referenced by [add\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [determinant\\_minor\(\)](#), [division\\_free\\_elimination\(\)](#), [echelon\\_form\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [imag\\_part\(\)](#), [is\\_zero\\_matrix\(\)](#), [let\\_op\(\)](#), [markowitz\\_elimination\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [op\(\)](#), [operator\(\)](#), [operator\(\)](#), [pivot\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [solve\(\)](#), [sub\(\)](#), [subs\(\)](#), [trace\(\)](#), and [transpose\(\)](#).

The documentation for this class was generated from the following files:

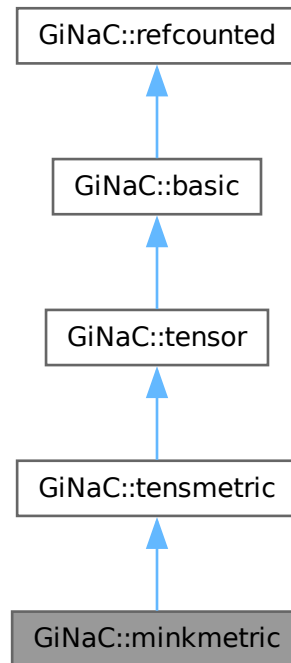
- [matrix.h](#)
- [matrix.cpp](#)

## 8.92 GiNaC::minkmetric Class Reference

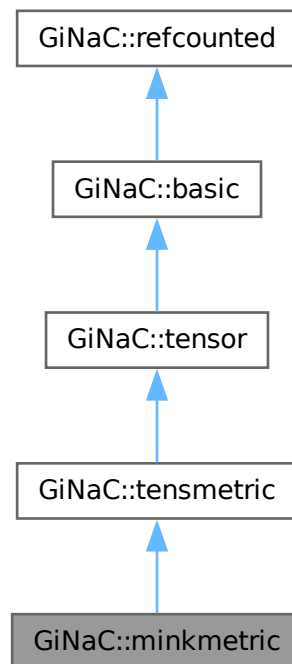
This class represents a Minkowski metric tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::minkmetric:



Collaboration diagram for `GiNaC::minkmetric`:



### Public Member Functions

- [minkmetric](#) (bool [pos\\_sig](#))  
*Construct Lorentz metric tensor with given signature.*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed Lorentz metric tensor.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*

### Public Member Functions inherited from [GiNaC::tensmetric](#)

- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed metric tensor with something else.*

## Public Member Functions inherited from GiNaC::tensor

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from GiNaC::basic

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const

- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &`s`) const  
*Return degree of highest power in object `s`.*
  - virtual int `ldegree` (const `ex` &`s`) const  
*Return degree of lowest power in object `s`.*
  - virtual `ex` `coeff` (const `ex` &`s`, int `n`=1) const  
*Return coefficient of degree `n` in object `s`.*
  - virtual `ex` `expand` (unsigned options=0) const  
*Expand expression, i.e.*
  - virtual `ex` `collect` (const `ex` &`s`, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
  - virtual `ex` `series` (const `relational` &`r`, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
  - virtual `ex` `normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const  
*Default implementation of `ex::normal()`.*
  - virtual `ex` `to_rational` (`exmap` &`repl`) const  
*Default implementation of `ex::to_rational()`.*
  - virtual `ex` `to_polynomial` (`exmap` &`repl`) const
  - virtual `numeric` `integer_content` () const
  - virtual `ex` `smod` (const `numeric` &`xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
  - virtual `numeric` `max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
  - virtual `exvector` `get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
  - virtual `ex` `add_indexed` (const `ex` &`self`, const `ex` &`other`) const  
*Add two indexed expressions.*
  - virtual `ex` `scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const  
*Multiply an indexed expression with a scalar.*
  - virtual `return_type_t` `return_type_tinfo` () const
  - virtual `ex` `conjugate` () const
  - virtual `ex` `real_part` () const
  - virtual `ex` `imag_part` () const
  - template<class `T`>  
void `print_dispatch` (const `print_context` &`c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
  - void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
  - `ex` `subs_one_level` (const `exmap` &`m`, unsigned options) const  
*Helper function for `subs()`.*
  - `ex` `diff` (const `symbol` &`s`, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
  - int `compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
  - bool `is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
  - const `basic` & `hold` () const  
*Stop further evaluation.*
  - unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned `f`) const  
*Set some `status_flags`.*
  - const `basic` & `clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensmetric](#)

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- bool [pos\\_sig](#)  
*If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 8.92.1 Detailed Description

This class represents a Minkowski metric tensor.

It has all the properties of a metric tensor and is (as a matrix) equal to  $\text{diag}(1,-1,-1,\dots)$  or  $\text{diag}(-1,1,1,\dots)$ .

### 8.92.2 Constructor & Destructor Documentation

#### 8.92.2.1 `minkmetric()`

```
GiNaC::minkmetric::minkmetric (
    bool pos_sig)
```

Construct Lorentz metric tensor with given signature.

References [minkmetric\(\)](#), and [pos\\_sig](#).

Referenced by [minkmetric\(\)](#).

### 8.92.3 Member Function Documentation

#### 8.92.3.1 `info()`

```
bool GiNaC::minkmetric::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).



### 8.92.3.2 eval\_indexed()

```
ex GiNaC::minkmetric::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed Lorentz metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), and [pos\\_sig](#).

### 8.92.3.3 archive()

```
void GiNaC::minkmetric::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_bool\(\)](#), and [pos\\_sig](#).

### 8.92.3.4 read\_archive()

```
void GiNaC::minkmetric::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::find\\_bool\(\)](#), and [pos\\_sig](#).

### 8.92.3.5 return\_type()

```
unsigned GiNaC::minkmetric::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.92.3.6 do\_print()

```
void GiNaC::minkmetric::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.92.3.7 do\_print\_latex()

```
void GiNaC::minkmetric::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

## 8.92.4 Member Data Documentation

### 8.92.4.1 pos\_sig

```
bool GiNaC::minkmetric::pos_sig [private]
```

If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .

Otherwise it is  $\text{diag}(1, -1, -1, \dots)$ .

Referenced by [archive\(\)](#), [eval\\_indexed\(\)](#), [minkmetric\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

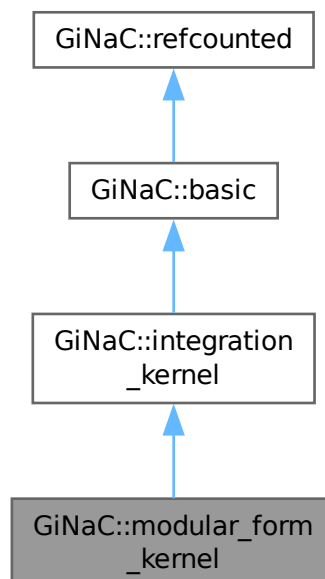
- [tensor.h](#)
- [tensor.cpp](#)

## 8.93 GiNaC::modular\_form\_kernel Class Reference

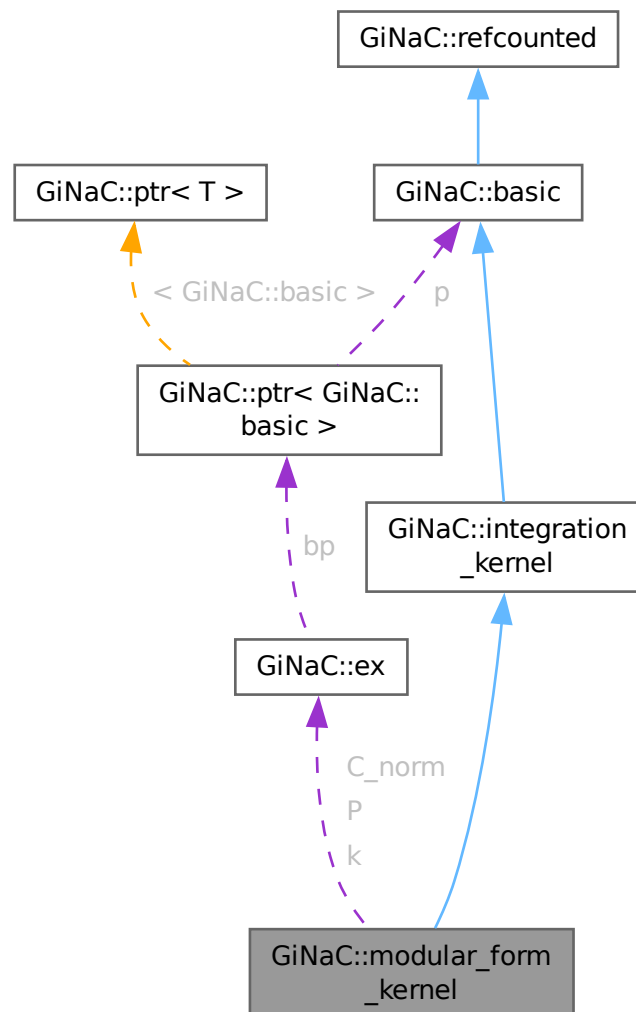
A kernel corresponding to a polynomial in Eisenstein series.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::modular\_form\_kernel:



Collaboration diagram for GiNaC::modular\_form\_kernel:



## Public Member Functions

- `modular_form_kernel` (const `ex` &`k`, const `ex` &`P`, const `ex` &`C_norm`=numeric(1))
- `ex series` (const `relational` &`r`, int order, unsigned options=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position  $i$ .*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position  $i$ .*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

- `ex Laurent_series` (const `ex` &qbar, int order) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &qbar, int N\_trunc=0) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &q, int order) const

## Public Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- `cln::cl_N series_coeff` (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const basic &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprntree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual `ex operator[]` (const `ex` &index) const

- virtual `ex operator[]` (size\_t i) const
- virtual `ex & operator[]` (const ex &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const ex &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const ex &pattern, exmap &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const exmap &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (map\_function &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const ex &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const ex &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const ex &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const ex &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const ex &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal` (exmap &repl, exmap &rev\_lookup, lst &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (exmap &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (exmap &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const numeric &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const ex &self, const ex &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const ex &self, const numeric &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, exvector &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>  
void `print_dispatch` (const print\_context &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const registered\_class\_info &ri, const print\_context &c, unsigned level) const

- Like `print()`, but dispatch to the specified class.
- virtual void `archive` (`archive_node` &n) const  
Save (serialize) the object into archive node.
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
Load (deserialize) the object from an archive node.
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
Helper function for `subs()`.
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
Default interface of `nth` derivative `ex::diff(s, n)`.
- int `compare` (const `basic` &other) const  
Compare objects syntactically to establish canonical ordering.
- bool `is_equal` (const `basic` &other) const  
Test for syntactic equality.
- const `basic` & `hold` () const  
Stop further evaluation.
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
Set some `status_flags`.
- const `basic` & `clearflag` (unsigned f) const  
Clear some `status_flags`.

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `uses_Laurent_series` () const override  
Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual `cln::cl_N series_coeff_impl` (int i) const  
For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
The actual implementation for computing a numerical value for the integrand.
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from GiNaC::basic

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex k](#)
- [ex P](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from GiNaC::integration\_kernel

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from GiNaC::basic

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.93.1 Detailed Description

A kernel corresponding to a polynomial in Eisenstein series.

This class represents the differential one-form

$$\omega^{\text{modular}}(P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)})) = C_k P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)}) \frac{d\bar{q}_N}{\bar{q}_N}.$$

## 8.93.2 Constructor & Destructor Documentation

### 8.93.2.1 modular\_form\_kernel()

```
GiNaC::modular_form_kernel::modular_form_kernel (
    const ex & k,
    const ex & P,
    const ex & C_norm = numeric(1))
```

References [C\\_norm](#), [k](#), [modular\\_form\\_kernel\(\)](#), and [P](#).

Referenced by [modular\\_form\\_kernel\(\)](#).

## 8.93.3 Member Function Documentation

### 8.93.3.1 series()

```
ex GiNaC::modular_form_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0) const \[override\], \[virtual\]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::relational::lhs\(\)](#), [P](#), [GiNaC::pow\(\)](#), [GiNaC::relational::rhs\(\)](#), and [GiNaC::ex::series\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 8.93.3.2 nops()

```
size_t GiNaC::modular_form_kernel::nops () const \[override\], \[virtual\]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.93.3.3 op()

```
ex GiNaC::modular_form_kernel::op (
    size_t i) const \[override\], \[virtual\]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), and [P](#).



#### 8.93.3.4 let\_op()

```
ex & GiNaC::modular_form_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), and [P](#).

#### 8.93.3.5 is\_numeric()

```
bool GiNaC::modular_form_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [k](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::series\\_to\\_poly\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 8.93.3.6 Laurent\_series()

```
ex GiNaC::modular_form_kernel::Laurent_series (
    const ex & x,
    int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::series\\_to\\_poly\(\)](#).

#### 8.93.3.7 get\_numerical\_value()

```
ex GiNaC::modular_form_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), and [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#).

### 8.93.3.8 uses\_Laurent\_series()

```
bool GiNaC::modular_form_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 8.93.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::modular_form_kernel::q_expansion_modular_form (
    const ex & q,
    int order) const
```

References [series\(\)](#).

Referenced by [is\\_numeric\(\)](#), and [Laurent\\_series\(\)](#).

### 8.93.3.10 do\_print()

```
void GiNaC::modular_form_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [C\\_norm](#), [k](#), [P](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.93.4 Member Data Documentation

### 8.93.4.1 k

```
ex GiNaC::modular_form_kernel::k [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [modular\\_form\\_kernel\(\)](#), and [op\(\)](#).

### 8.93.4.2 P

```
ex GiNaC::modular_form_kernel::P [protected]
```

Referenced by [do\\_print\(\)](#), [let\\_op\(\)](#), [modular\\_form\\_kernel\(\)](#), [op\(\)](#), and [series\(\)](#).

### 8.93.4.3 C\_norm

`ex` GiNaC::modular\_form\_kernel::C\_norm [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), [modular\\_form\\_kernel\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.94 GiNaC::basic\_partition\_generator::mpartition2 Struct Reference

```
#include <utils.h>
```

### Public Member Functions

- [mpartition2](#) (unsigned n\_, unsigned m\_)
- bool [next\\_partition](#) ()

### Public Attributes

- std::vector< unsigned > [x](#)
- unsigned [n](#)
- unsigned [m](#)

## 8.94.1 Constructor & Destructor Documentation

### 8.94.1.1 mpartition2()

```
GiNaC::basic_partition_generator::mpartition2::mpartition2 (
    unsigned n_,
    unsigned m_) [inline]
```

References [m](#), [n](#), and [x](#).

## 8.94.2 Member Function Documentation

### 8.94.2.1 next\_partition()

```
bool GiNaC::basic_partition_generator::mpartition2::next_partition () [inline]
```

References [m](#), and [x](#).

### 8.94.3 Member Data Documentation

#### 8.94.3.1 `x`

```
std::vector<unsigned> GiNaC::basic_partition_generator::mpartition2::x
```

Referenced by [mpartition2\(\)](#), and [next\\_partition\(\)](#).

#### 8.94.3.2 `n`

```
unsigned GiNaC::basic_partition_generator::mpartition2::n
```

Referenced by [mpartition2\(\)](#).

#### 8.94.3.3 `m`

```
unsigned GiNaC::basic_partition_generator::mpartition2::m
```

Referenced by [mpartition2\(\)](#), and [next\\_partition\(\)](#).

The documentation for this struct was generated from the following file:

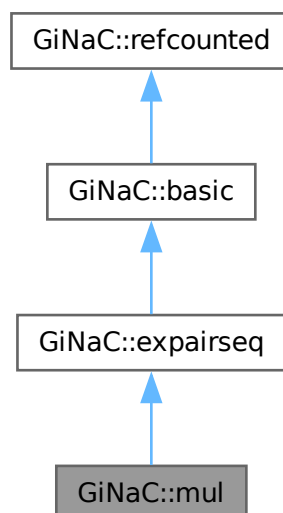
- [utils.h](#)

## 8.95 GiNaC::mul Class Reference

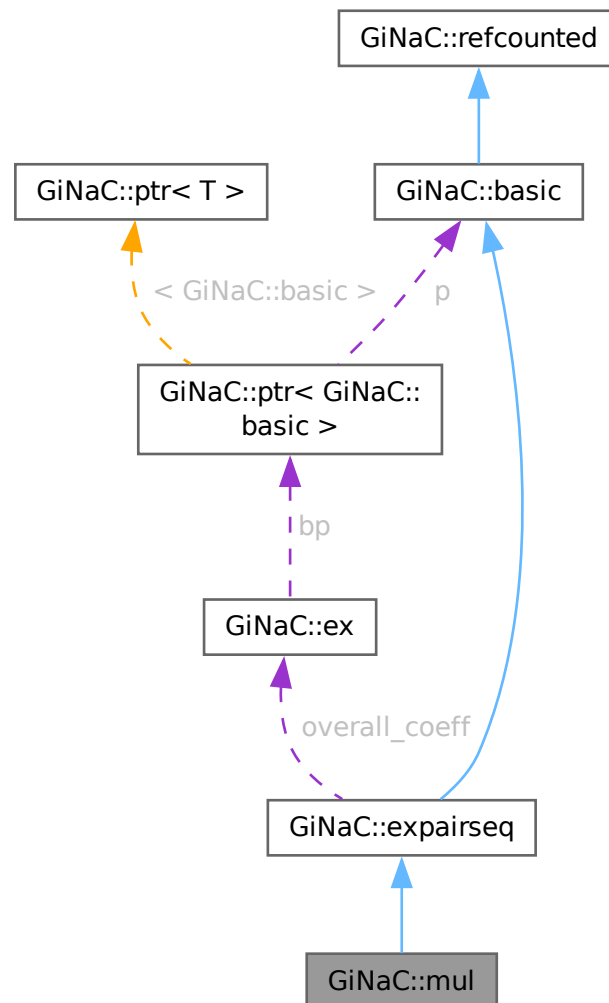
Product of expressions.

```
#include <mul.h>
```

Inheritance diagram for GiNaC::mul:



Collaboration diagram for GiNaC::mul:



## Public Member Functions

- `mul` (const `ex` &lh, const `ex` &rh)
- `mul` (const `exvector` &v)
- `mul` (const `epvector` &v)
- `mul` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (`epvector` &&vp)
- `mul` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (const `ex` &lh, const `ex` &mh, const `ex` &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override

- *Check whether this is a polynomial in the given variables.*
- `int degree (const ex &s) const` override  
*Return degree of highest power in object s.*
- `int ldegree (const ex &s) const` override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override  
*Return coefficient of degree n in object s.*
- `bool has (const ex &other, unsigned options=0) const` override  
*Test for occurrence of a pattern.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalf ()` const override  
*Evaluate object numerically.*
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &s, int order, unsigned options=0) const` override  
*Implementation of [ex::series\(\)](#) for product.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override  
*Implementation of [ex::normal\(\)](#) for a product.*
- `numeric integer_content ()` const override
- `ex smod (const numeric &xi) const` override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient ()` const override  
*Implementation [ex::max\\_coefficient\(\)](#).*
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- `ex conjugate ()` const override
- `ex algebraic_subs_mul (const exmap &m, unsigned options) const`

## Public Member Functions inherited from [GiNaC::expairseq](#)

- `expairseq (const ex &lh, const ex &rh)`
- `expairseq (const exvector &v)`
- `expairseq (const epvector &v, const ex &oc, bool do_index_renaming=false)`
- `expairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false)`
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i) const` override  
*Return operand/member at position i.*
- `ex map (map\_function &f) const` override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex to_rational (exmap &repl) const` override  
*Implementation of [ex::to\\_rational\(\)](#) for expairseqs.*
- `ex to_polynomial (exmap &repl) const` override  
*Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.*
- `bool match (const ex &pattern, exmap &repl_lst) const` override  
*Check whether the expression matches a given pattern.*
- `ex subs (const exmap &m, unsigned options=0) const` override  
*Substitute a set of objects by arbitrary expressions.*
- `void archive (archive\_node &n) const` override  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive\_node &n, lst &symbols) const` override  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual void `accept` (GiNaC::visitor &v) const
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for subs().*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative ex::diff(s, n).*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some status\_flags.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some status\_flags.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a product.*
- `ex eval_ncmul` (const `exvector` &v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false) const override  
*Create an object of this type.*
- `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const override
- `expair split_ex_to_pair` (const `ex` &e) const override  
*Form an expair from an ex, using the corresponding semantics.*
- `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const override
- `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const override
- `ex recombine_pair_to_ex` (const `expair` &p) const override  
*Form an ex out of an expair, using the corresponding semantics.*
- bool `expair_needs_further_processing` (`ep` it) override
- `ex default_overall_coeff` () const override
- void `combine_overall_coeff` (const `ex` &c) override
- void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2) override
- bool `can_make_flat` (const `expair` &p) const override
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- void `find_real_imag` (`ex` &, `ex` &) const
- void `print_overall_coeff` (const `print_context` &c, const char \*mul\_sym) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `epvector expandchildren` (unsigned options) const  
*Member-wise expand the expairs representing this sequence.*

## Protected Member Functions inherited from `GiNaC::expairseq`

- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)



- void [construct\\_from\\_2\\_expireseq](#) (const [expireseq](#) &s1, const [expireseq](#) &s2)
- void [construct\\_from\\_expireseq\\_ex](#) (const [expireseq](#) &s, const [ex](#) &e)
- void [construct\\_from\\_exvector](#) (const [exvector](#) &v)
- void [construct\\_from\\_epvector](#) (const [epvector](#) &v, bool do\_index\_renaming=false)
- void [construct\\_from\\_epvector](#) ([epvector](#) &&v, bool do\_index\_renaming=false)
- void [make\\_flat](#) (const [exvector](#) &v)  
*Combine this expireseq with argument exvector.*
- void [make\\_flat](#) (const [epvector](#) &v, bool do\_index\_renaming=false)  
*Combine this expireseq with argument epvector.*
- void [canonicalize](#) ()  
*Brings this expireseq into a sorted (canonical) form.*
- void [combine\\_same\\_terms\\_sorted\\_seq](#) ()  
*Compact a presorted expireseq by combining all matching expires to one each.*
- bool [is\\_canonical](#) () const  
*Check if this expireseq is in sorted (canonical) form.*
- [epvector](#) [expandchildren](#) (unsigned options) const  
*Member-wise expand the expires in this sequence.*
- [epvector](#) [evalchildren](#) () const  
*Member-wise evaluate the expires in this sequence.*
- [epvector](#) [subschildren](#) (const [exmap](#) &m, unsigned options=0) const  
*Member-wise substitute in this sequence.*

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Static Protected Member Functions

- static bool [can\\_be\\_further\\_expanded](#) (const [ex](#) &e)

## Friends

- class [add](#)
- class [ncmul](#)
- class [power](#)

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::expairseq](#)

- [epvector seq](#)
- [ex overall\\_coeff](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 8.95.1 Detailed Description

Product of expressions.

## 8.95.2 Constructor & Destructor Documentation

### 8.95.2.1 [mul\(\)](#) [1/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & rh)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [do\\_print\\_latex\(\)](#), [expand\(\)](#), [power](#), and [smod\(\)](#).

### 8.95.2.2 [mul\(\)](#) [2/7]

```
GiNaC::mul::mul (
    const exvector & v)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 8.95.2.3 [mul\(\)](#) [3/7]

```
GiNaC::mul::mul (
    const epvector & v)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.95.2.4 mul()** [4/7]

```
GiNaC::mul::mul (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.95.2.5 mul()** [5/7]

```
GiNaC::mul::mul (
    epvector && vp)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.95.2.6 mul()** [6/7]

```
GiNaC::mul::mul (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.95.2.7 mul()** [7/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & mh,
    const ex & rh)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**8.95.3 Member Function Documentation****8.95.3.1 precedence()**

```
unsigned GiNaC::mul::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::expairseq](#).

References [precedence\(\)](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), [precedence\(\)](#), and [print\\_overall\\_coeff\(\)](#).

### 8.95.3.2 info()

```
bool GiNaC::mul::info (
    unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::factor\(\)](#), [GiNaC::basic::flags](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::status\\_flags::is\\_negative](#), [GiNaC::status\\_flags::is\\_positive](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), and [info\(\)](#).

### 8.95.3.3 is\_polynomial()

```
bool GiNaC::mul::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.4 degree()

```
int GiNaC::mul::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::degree\(\)](#), [GiNaC::ex\\_to\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.5 ldegree()

```
int GiNaC::mul::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::ldegree\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.6 coeff()

```
ex GiNaC::mul::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [coeff\(\)](#), and [print\\_overall\\_coeff\(\)](#).

### 8.95.3.7 has()

```
bool GiNaC::mul::has (
    const ex & pattern,
    unsigned options = 0) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::is\\_a\(\)](#), and [GiNaC::expairseq::nops\(\)](#).

### 8.95.3.8 eval()

```
ex GiNaC::mul::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x$ ,  $x_1$ ,  $x_2$ ,... stand for a symbolic variables of type `ex` and  $c$ ,  $c_1$ ,  $c_2$ ... stand for such expressions that contain a plain number.

- $(...,x;0) \rightarrow 0$
- $*(+(x_1,x_2,...);c) \rightarrow *(*(x_1,c),(x_2,c),...)$
- $*(x;1) \rightarrow x$
- $*(;c) \rightarrow c$

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [add](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::integer\\_content](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [likely](#), [GiNaC::numeric::mul\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

### 8.95.3.9 evalf()

```
ex GiNaC::mul::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.10 real\_part()

```
ex GiNaC::mul::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).

### 8.95.3.11 imag\_part()

```
ex GiNaC::mul::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).

### 8.95.3.12 evalm()

```
ex GiNaC::mul::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::evalm\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.95.3.13 series()

```
ex GiNaC::mul::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for product.

This performs series multiplication when multiplying series.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 8.95.3.14 normal()

```
ex GiNaC::mul::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a product.

It cancels common factors from fractions.

See also

[ex::normal\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.15 integer\_content()

```
numeric GiNaC::mul::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 8.95.3.16 smod()

```
ex GiNaC::mul::smod (
    const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

#### Parameters

$xi$	modulus
------	---------

#### Returns

mapped polynomial

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::is\\_exactly\\_a\(\)](#), [mul\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

**8.95.3.17 max\_coefficient()**

```
numeric GiNaC::mul::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

**8.95.3.18 get\_free\_indices()**

```
exvector GiNaC::mul::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**8.95.3.19 conjugate()**

```
ex GiNaC::mul::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

**8.95.3.20 derivative()**

```
ex GiNaC::mul::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::diff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::expair::swap\(\)](#).



**8.95.3.21 eval\_ncmul()**

```
ex GiNaC::mul::eval_ncmul (
    const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**8.95.3.22 return\_type()**

```
unsigned GiNaC::mul::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_com](#) and [GiNaC::expairseq::seq](#).

**8.95.3.23 return\_type\_tinfo()**

```
return\_type\_t GiNaC::mul::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::make\\_return\\_type\\_t\(\)](#), [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**8.95.3.24 thisexpairseq() [1/2]**

```
ex GiNaC::mul::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::dynallocate\(\)](#).

Referenced by [conjugate\(\)](#).

**8.95.3.25 thisexpairseq() [2/2]**

```
ex GiNaC::mul::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::dynallocate\(\)](#).

### 8.95.3.26 split\_ex\_to\_pair()

```
expair GiNaC::mul::split_ex_to_pair (
    const ex & e) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::power::basis](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::power::exponent](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [power](#).

Referenced by [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expand\(\)](#), and [expandchildren\(\)](#).

### 8.95.3.27 combine\_ex\_with\_coeff\_to\_pair()

```
expair GiNaC::mul::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::pow\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.95.3.28 combine\_pair\_with\_coeff\_to\_pair()

```
expair GiNaC::mul::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::pow\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expair::rest](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [GiNaC::power::expand\\_mul\(\)](#).

### 8.95.3.29 recombine\_pair\_to\_ex()

```
ex GiNaC::mul::recombine_pair_to_ex (
    const expair & p) const [override], [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [coeff\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::power::expand\(\)](#), [expandchildren\(\)](#), [find\\_real\\_imag\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [ldegree\(\)](#), [max\\_coefficient\(\)](#), [normal\(\)](#), [series\(\)](#), and [smod\(\)](#).

### 8.95.3.30 `expair_needs_further_processing()`

```
bool GiNaC::mul::expair_needs_further_processing (
    exp it) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.95.3.31 `default_overall_coeff()`

```
ex GiNaC::mul::default_overall_coeff () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#).

### 8.95.3.32 `combine_overall_coeff()` [1/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 8.95.3.33 `combine_overall_coeff()` [2/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c1,
    const ex & c2) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [power](#).

### 8.95.3.34 `can_make_flat()`

```
bool GiNaC::mul::can_make_flat (
    const expair & p) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

### 8.95.3.35 expand()

```
ex GiNaC::mul::expand (
    unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num0\\_p](#), [add](#), [can\\_be\\_further\\_expanded\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_rename\\_idx](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [mul\(\)](#), [GiNaC::numeric::mul\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 8.95.3.36 algebraic\_subs\_mul()

```
ex GiNaC::mul::algebraic_subs_mul (
    const exmap & m,
    unsigned options) const
```

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

### 8.95.3.37 find\_real\_imag()

```
void GiNaC::mul::find_real_imag (
    ex & rp,
    ex & ip) const [protected]
```

References [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 8.95.3.38 print\_overall\_coeff()

```
void GiNaC::mul::print_overall_coeff (
    const print_context & c,
    const char * mul_sym) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [coeff\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**8.95.3.39 do\_print()**

```
void GiNaC::mul::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [GiNaC::class\\_info< OPT >::first](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::expairseq::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.95.3.40 do\_print\_latex()**

```
void GiNaC::mul::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [mul\(\)](#), [precedence\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::expairseq::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.95.3.41 do\_print\_csrc()**

```
void GiNaC::mul::do_print_csrc (
    const print\_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::expairseq::overall\\_coeff](#), [power](#), [precedence\(\)](#), [GiNaC::print\\_context::s](#), and [GiNaC::expairseq::seq](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.95.3.42 do\_print\_python\_repr()**

```
void GiNaC::mul::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.95.3.43 can\_be\_further\_expanded()**

```
bool GiNaC::mul::can_be_further_expanded (
    const ex & e) [static], [protected]
```

References [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::posint](#), and [GiNaC::expairseq::seq](#).

Referenced by [expand\(\)](#).

### 8.95.3.44 `expandchildren()`

```
epvector GiNaC::mul::expandchildren (
    unsigned options) const [protected]
```

Member-wise expand the expairs representing this sequence.

This must be overridden from `expairseq::expandchildren()` and done iteratively in order to allow for early cancellations and thus safe memory.

See also

[mul::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [expand\(\)](#).

## 8.95.4 Friends And Related Symbol Documentation

### 8.95.4.1 `add`

```
friend class add [friend]
```

References [add](#).

Referenced by [add](#), [eval\(\)](#), and [expand\(\)](#).

### 8.95.4.2 `ncmul`

```
friend class ncmul [friend]
```

References [ncmul](#).

Referenced by [ncmul](#).

### 8.95.4.3 `power`

```
friend class power [friend]
```

References [mul\(\)](#), and [power](#).

Referenced by [combine\\_overall\\_coeff\(\)](#), [do\\_print\\_csrc\(\)](#), [power](#), and [split\\_ex\\_to\\_pair\(\)](#).

The documentation for this class was generated from the following files:

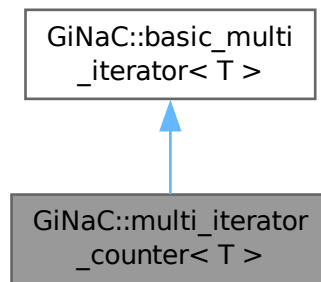
- [mul.h](#)
- [indexed.cpp](#)
- [mul.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 8.96 GiNaC::multi\_iterator\_counter< T > Class Template Reference

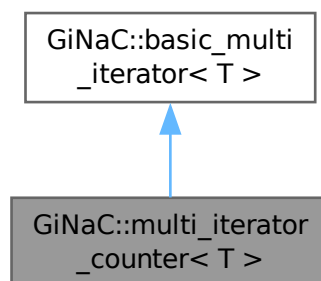
The class `multi_iterator_counter` defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_counter< T >:



Collaboration diagram for GiNaC::multi\_iterator\_counter< T >:



### Public Member Functions

- `multi_iterator_counter` (void)  
*Default constructor.*
- `multi_iterator_counter` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_counter` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > & init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > & operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_counter](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

### 8.96.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_counter< T >
```

The class [multi\\_iterator\\_counter](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$



## 8.96.2 Constructor & Destructor Documentation

### 8.96.2.1 multi\_iterator\_counter() [1/3]

```
template<class T>
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#).

Referenced by [operator<<](#).

### 8.96.2.2 multi\_iterator\_counter() [2/3]

```
template<class T>
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::N](#).

### 8.96.2.3 multi\_iterator\_counter() [3/3]

```
template<class T>
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.96.3 Member Function Documentation

### 8.96.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

### 8.96.3.2 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.96.4 Friends And Related Symbol Documentation

### 8.96.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_counter< TT > & v) [friend]
```

References [multi\\_iterator\\_counter\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

The documentation for this class was generated from the following file:

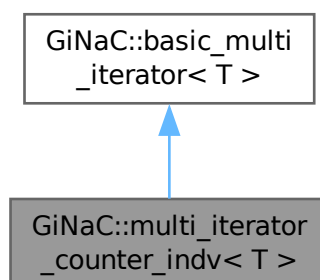
- [utils\\_multi\\_iterator.h](#)

## 8.97 GiNaC::multi\_iterator\_counter\_indv< T > Class Template Reference

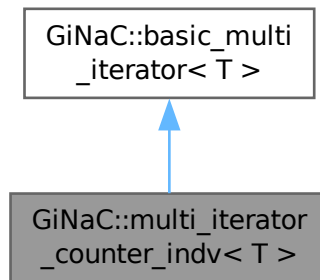
The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#):



Collaboration diagram for GiNaC::multi\_iterator\_counter\_indv< T >:



### Public Member Functions

- [multi\\_iterator\\_counter\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_counter\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_counter\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator< T > & init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator< T > & operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)

- Subscription via `[]`.
- `T operator() (size_t i) const`  
Subscription via `()`.
- `T & operator() (size_t i)`  
Subscription via `()`.

### Protected Attributes

- `std::vector< T > Nv`

### Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

### Friends

- `template<class TT>`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_counter_indv< TT > &v)`

## 8.97.1 Detailed Description

`template<class T>`  
`class GiNaC::multi_iterator_counter_indv< T >`

The class `multi_iterator_counter_indv` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

## 8.97.2 Constructor & Destructor Documentation

### 8.97.2.1 `multi_iterator_counter_indv()` [1/3]

```
template<class T>
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    void ) [inline]
```

Default constructor.

References `GiNaC::basic_multi_iterator< T >::basic_multi_iterator()`, and `Nv`.

Referenced by `operator<<`.

**8.97.2.2 multi\_iterator\_counter\_indv()** [2/3]

```
template<class T>
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [Nv](#).

**8.97.2.3 multi\_iterator\_counter\_indv()** [3/3]

```
template<class T>
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [Nv](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

**8.97.3 Member Function Documentation****8.97.3.1 init()**

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

**8.97.3.2 operator++()**

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [Nv](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >](#)

## 8.97.4 Friends And Related Symbol Documentation

### 8.97.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< TT > & v) [friend]
```

References [multi\\_iterator\\_counter\\_indv\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.97.5 Member Data Documentation

### 8.97.5.1 Nv

```
template<class T>
std::vector<T> GiNaC::multi_iterator_counter_indv< T >::Nv [protected]
```

Referenced by [multi\\_iterator\\_counter\\_indv\(\)](#), [multi\\_iterator\\_counter\\_indv\(\)](#), [multi\\_iterator\\_counter\\_indv\(\)](#), and [operator++\(\)](#).

The documentation for this class was generated from the following file:

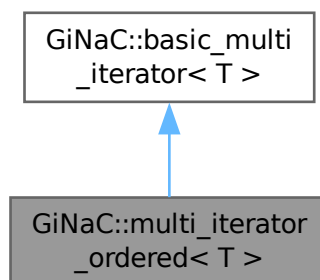
- [utils\\_multi\\_iterator.h](#)

## 8.98 GiNaC::multi\_iterator\_ordered< T > Class Template Reference

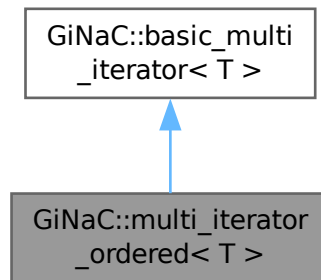
The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for [GiNaC::multi\\_iterator\\_ordered< T >](#):



Collaboration diagram for GiNaC::multi\_iterator\_ordered< T >:



### Public Member Functions

- `multi_iterator_ordered` (void)  
*Default constructor.*
- `multi_iterator_ordered` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- `multi_iterator_ordered` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Public Member Functions inherited from `GiNaC::basic_multi_iterator< T >`

- `basic_multi_iterator` (void)  
*Default constructor.*
- `basic_multi_iterator` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- `basic_multi_iterator` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual `~basic_multi_iterator` ()  
*Destructor.*
- size\_t `size` (void) const  
*Returns the size of a multi\_iterator.*
- bool `overflow` (void) const  
*Return the overflow flag.*
- const std::vector< T > &`get_vector` (void) const  
*Returns a reference to the vector v.*
- T `operator[]` (size\_t i) const  
*Subscription via [].*
- T &`operator[]` (size\_t i)

- *Subscription via [].*  
• T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

### 8.98.1 Detailed Description

template<class T>  
class [GiNaC::multi\\_iterator\\_ordered](#)< T >

The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .

### 8.98.2 Constructor & Destructor Documentation

#### 8.98.2.1 [multi\\_iterator\\_ordered\(\)](#) [1/3]

```
template<class T>
GiNaC::multi\_iterator\_ordered< T >::multi_iterator_ordered (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#).

Referenced by [operator<<](#).



**8.98.2.2 multi\_iterator\_ordered()** [2/3]

```
template<class T>
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::N](#).

**8.98.2.3 multi\_iterator\_ordered()** [3/3]

```
template<class T>
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

**8.98.3 Member Function Documentation****8.98.3.1 init()**

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

**8.98.3.2 operator++()**

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.98.4 Friends And Related Symbol Documentation

### 8.98.4.1 `operator<<`

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered< TT > & v) [friend]
```

References [multi\\_iterator\\_ordered\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

The documentation for this class was generated from the following file:

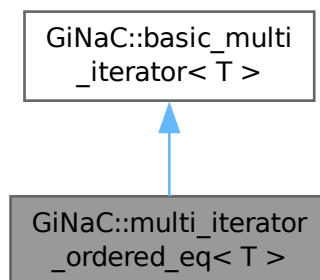
- [utils\\_multi\\_iterator.h](#)

## 8.99 GiNaC::multi\_iterator\_ordered\_eq< T > Class Template Reference

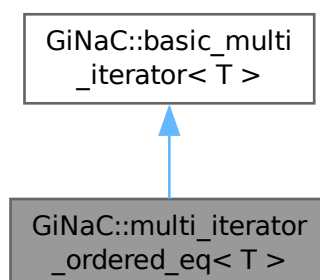
The class [multi\\_iterator\\_ordered\\_eq](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered\_eq< T >:



Collaboration diagram for GiNaC::multi\_iterator\_ordered\_eq< T >:



## Public Member Functions

- [multi\\_iterator\\_ordered\\_eq](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator](#)< T >

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator](#)< T >

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

### 8.99.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered_eq< T >
```

The class [multi\\_iterator\\_ordered\\_eq](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .

### 8.99.2 Constructor & Destructor Documentation

#### 8.99.2.1 multi\_iterator\_ordered\_eq() [1/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#).

Referenced by [operator<<](#).

#### 8.99.2.2 multi\_iterator\_ordered\_eq() [2/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::N](#).

#### 8.99.2.3 multi\_iterator\_ordered\_eq() [3/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

### 8.99.3 Member Function Documentation

#### 8.99.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

#### 8.99.3.2 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

### 8.99.4 Friends And Related Symbol Documentation

#### 8.99.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< TT > & v) [friend]
```

References [multi\\_iterator\\_ordered\\_eq\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

The documentation for this class was generated from the following file:

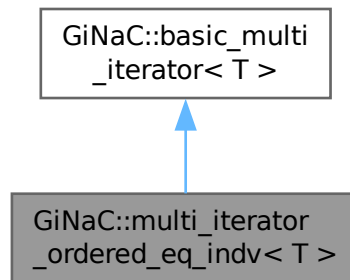
- [utils\\_multi\\_iterator.h](#)

## 8.100 GiNaC::multi\_iterator\_ordered\_eq\_indv< T > Class Template Reference

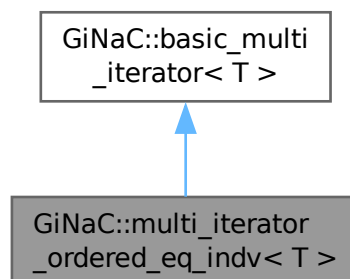
The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



Collaboration diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



### Public Member Functions

- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

**Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

**Protected Attributes**

- std::vector< T > [Nv](#)

**Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

**Friends**

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq\\_indv](#)< TT > &v)

**8.100.1 Detailed Description**

template<class T>  
class [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv](#)< T >

The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

and

$$i_j \leq i_{j+1}.$$

## 8.100.2 Constructor & Destructor Documentation

### 8.100.2.1 multi\_iterator\_ordered\_eq\_indv() [1/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [Nv](#).

Referenced by [operator<<](#).

### 8.100.2.2 multi\_iterator\_ordered\_eq\_indv() [2/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [Nv](#).

### 8.100.2.3 multi\_iterator\_ordered\_eq\_indv() [3/3]

```
template<class T>
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [Nv](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.100.3 Member Function Documentation

### 8.100.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).



### 8.100.3.2 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [Nv](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.100.4 Friends And Related Symbol Documentation

### 8.100.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< TT > & v) [friend]
```

References [multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.100.5 Member Data Documentation

### 8.100.5.1 Nv

```
template<class T>
std::vector<T> GiNaC::multi_iterator_ordered_eq_indv< T >::Nv [protected]
```

Referenced by [multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), [multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), [multi\\_iterator\\_ordered\\_eq\\_indv\(\)](#), and [operator++\(\)](#).

The documentation for this class was generated from the following file:

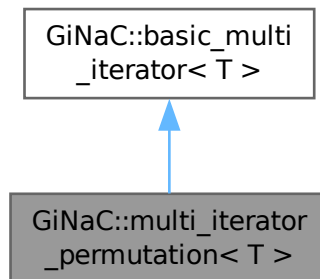
- [utils\\_multi\\_iterator.h](#)

## 8.101 GiNaC::multi\_iterator\_permutation< T > Class Template Reference

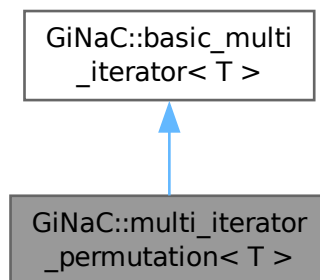
The class `multi_iterator_permutation` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , for which.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_permutation< T >`:



Collaboration diagram for `GiNaC::multi_iterator_permutation< T >`:



### Public Member Functions

- `multi_iterator_permutation` (void)  
*Default constructor.*
- `multi_iterator_permutation` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_permutation` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*

- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.*
- int [get\\_sign](#) (void) const  
*Returns the sign of the permutation, defined by.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator](#)< T >

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT>  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_permutation](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator](#)< T >

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

### 8.101.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_permutation< T >
```

The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

### 8.101.2 Constructor & Destructor Documentation

#### 8.101.2.1 multi\_iterator\_permutation() [1/3]

```
template<class T>
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#).

Referenced by [operator<<](#).

#### 8.101.2.2 multi\_iterator\_permutation() [2/3]

```
template<class T>
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::N](#).

#### 8.101.2.3 multi\_iterator\_permutation() [3/3]

```
template<class T>
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

### 8.101.3 Member Function Documentation

#### 8.101.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

#### 8.101.3.2 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::N](#), [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

#### 8.101.3.3 get\_sign()

```
template<class T>
int GiNaC::multi_iterator_permutation< T >::get_sign (
    void ) const [inline]
```

Returns the sign of the permutation, defined by.

$$(-1)^{n_{inv}},$$

where  $n_{inv}$  is the number of inversions, e.g. the number of pairs  $i < j$  for which

$$n_i > n_j.$$

References [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.101.4 Friends And Related Symbol Documentation

### 8.101.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_permutation< TT > & v) [friend]
```

References [multi\\_iterator\\_permutation\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

The documentation for this class was generated from the following file:

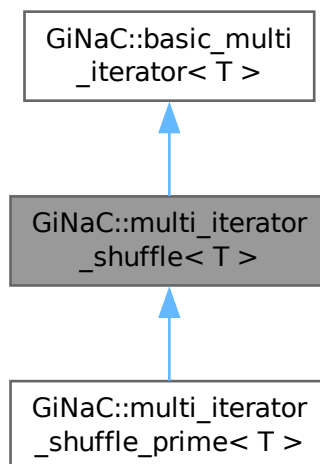
- [utils\\_multi\\_iterator.h](#)

## 8.102 GiNaC::multi\_iterator\_shuffle< T > Class Template Reference

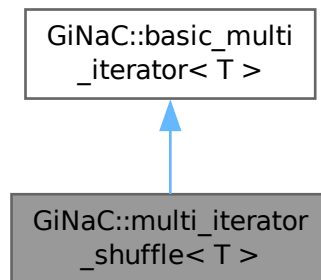
The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle< T >:



Collaboration diagram for GiNaC::multi\_iterator\_shuffle< T >:



### Public Member Functions

- [multi\\_iterator\\_shuffle](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to the first shuffle.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

**Protected Attributes**

- `size_t` [N\\_internal](#)
- `std::vector< size_t >` [v\\_internal](#)
- `std::vector< T >` [v\\_orig](#)

**Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- `T` [N](#)
- `T` [B](#)
- `std::vector< T >` [v](#)
- `bool` [flag\\_overflow](#)

**Friends**

- `template<class TT>`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle< TT > &v)`

**8.102.1 Detailed Description**

`template<class T>`  
**class** [GiNaC::multi\\_iterator\\_shuffle< T >](#)

The class [multi\\_iterator\\_shuffle](#) defines a `multi_iterator`, which runs over all shuffles of `a` and `b`.

**8.102.2 Constructor & Destructor Documentation****8.102.2.1 [multi\\_iterator\\_shuffle\(\)](#) [1/2]**

```
template<class T>
GiNaC::multi\_iterator\_shuffle< T >::multi\_iterator\_shuffle (
    void ) [inline]
```

Default constructor.

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [N\\_internal](#), [v\\_internal](#), and [v\\_orig](#).

Referenced by [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::multi\\_iterator\\_shuffle\\_prime\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::](#) and [operator<<](#).

**8.102.2.2 [multi\\_iterator\\_shuffle\(\)](#) [2/2]**

```
template<class T>
GiNaC::multi\_iterator\_shuffle< T >::multi\_iterator\_shuffle (
    const std::vector< T > & a,
    const std::vector< T > & b) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [N\\_internal](#), [GiNaC::basic\\_multi\\_iterator< T >::v](#), [v\\_internal](#), and [v\\_orig](#).



## 8.102.3 Member Function Documentation

### 8.102.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::v](#), [v\\_internal](#), and [v\\_orig](#).

Referenced by [GiNaC::Gt::regularise\(\)](#).

### 8.102.3.2 operator++()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::basic\\_multi\\_iterator\(\)](#), [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [N\\_internal](#), [GiNaC::basic\\_multi\\_iterator< T >::v](#), [v\\_internal](#), and [v\\_orig](#).

## 8.102.4 Friends And Related Symbol Documentation

### 8.102.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< TT > & v) [friend]
```

References [multi\\_iterator\\_shuffle\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

## 8.102.5 Member Data Documentation

### 8.102.5.1 N\_internal

```
template<class T>
size_t GiNaC::multi_iterator_shuffle< T >::N_internal [protected]
```

Referenced by [multi\\_iterator\\_shuffle\(\)](#), [multi\\_iterator\\_shuffle\(\)](#), and [operator++\(\)](#).

### 8.102.5.2 v\_internal

```
template<class T>
std::vector<size_t> GiNaC::multi_iterator_shuffle< T >::v_internal [protected]
```

Referenced by [init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::init\(\)](#), [multi\\_iterator\\_shuffle\(\)](#), [multi\\_iterator\\_shuffle\(\)](#), and [operator++\(\)](#).

### 8.102.5.3 v\_orig

```
template<class T>
std::vector<T> GiNaC::multi_iterator_shuffle< T >::v_orig [protected]
```

Referenced by [init\(\)](#), [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >::init\(\)](#), [multi\\_iterator\\_shuffle\(\)](#), [multi\\_iterator\\_shuffle\(\)](#), and [operator++\(\)](#).

The documentation for this class was generated from the following file:

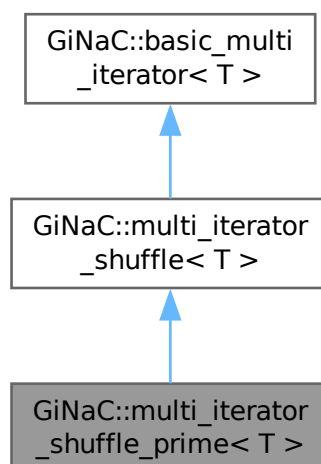
- [utils\\_multi\\_iterator.h](#)

## 8.103 GiNaC::multi\_iterator\_shuffle\_prime< T > Class Template Reference

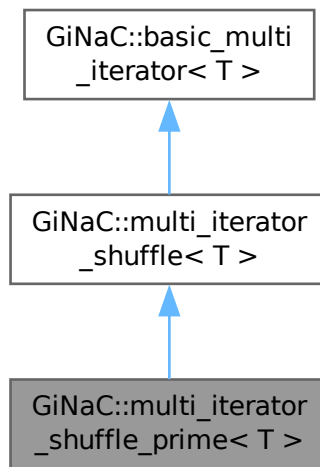
The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



Collaboration diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



### Public Member Functions

- `multi_iterator_shuffle_prime` (void)  
*Default constructor.*
- `multi_iterator_shuffle_prime` (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to the first shuffle.*

### Public Member Functions inherited from `GiNaC::multi_iterator_shuffle< T >`

- `multi_iterator_shuffle` (void)  
*Default constructor.*
- `multi_iterator_shuffle` (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.*

### Public Member Functions inherited from `GiNaC::basic_multi_iterator< T >`

- `basic_multi_iterator` (void)  
*Default constructor.*
- `basic_multi_iterator` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- `basic_multi_iterator` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual `~basic_multi_iterator` ()

*Destructor.*

- `size_t size` (void) const  
*Returns the size of a multi\_iterator.*
- `bool overflow` (void) const  
*Return the overflow flag.*
- `const std::vector< T > & get_vector` (void) const  
*Returns a reference to the vector v.*
- `T operator[]` (size\_t i) const  
*Subscription via [].*
- `T & operator[]` (size\_t i)  
*Subscription via [].*
- `T operator()` (size\_t i) const  
*Subscription via ().*
- `T & operator()` (size\_t i)  
*Subscription via ().*

## Friends

- `template<class TT>`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_shuffle_prime< TT > &v)`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::multi_iterator_shuffle< T >`

- `size_t N_internal`
- `std::vector< size_t > v_internal`
- `std::vector< T > v_orig`

### Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

## 8.103.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle_prime< T >
```

The class `multi_iterator_shuffle_prime` defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

## 8.103.2 Constructor & Destructor Documentation

### 8.103.2.1 multi\_iterator\_shuffle\_prime() [1/2]

```
template<class T>
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    void ) [inline]
```

Default constructor.

References [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

Referenced by [operator<<](#).

### 8.103.2.2 multi\_iterator\_shuffle\_prime() [2/2]

```
template<class T>
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    const std::vector< T > & a,
    const std::vector< T > & b) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

## 8.103.3 Member Function Documentation

### 8.103.3.1 init()

```
template<class T>
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle_prime< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::multi\\_iterator\\_shuffle< T >](#).

References [GiNaC::basic\\_multi\\_iterator< T >::flag\\_overflow](#), [GiNaC::basic\\_multi\\_iterator< T >::v](#), [GiNaC::multi\\_iterator\\_shuffle< T >](#) and [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_orig](#).

## 8.103.4 Friends And Related Symbol Documentation

### 8.103.4.1 operator<<

```
template<class T>
template<class TT>
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< TT > & v) [friend]
```

References [multi\\_iterator\\_shuffle\\_prime\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::v](#).

The documentation for this class was generated from the following file:

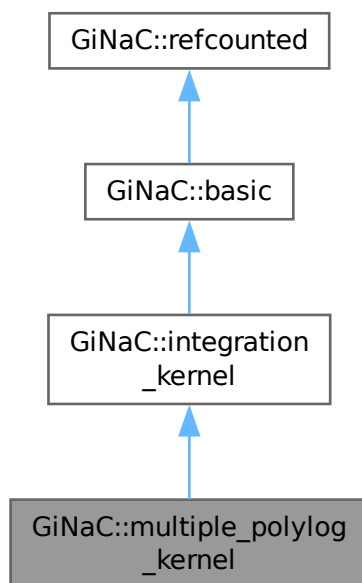
- [utils\\_multi\\_iterator.h](#)

## 8.104 GiNaC::multiple\_polylog\_kernel Class Reference

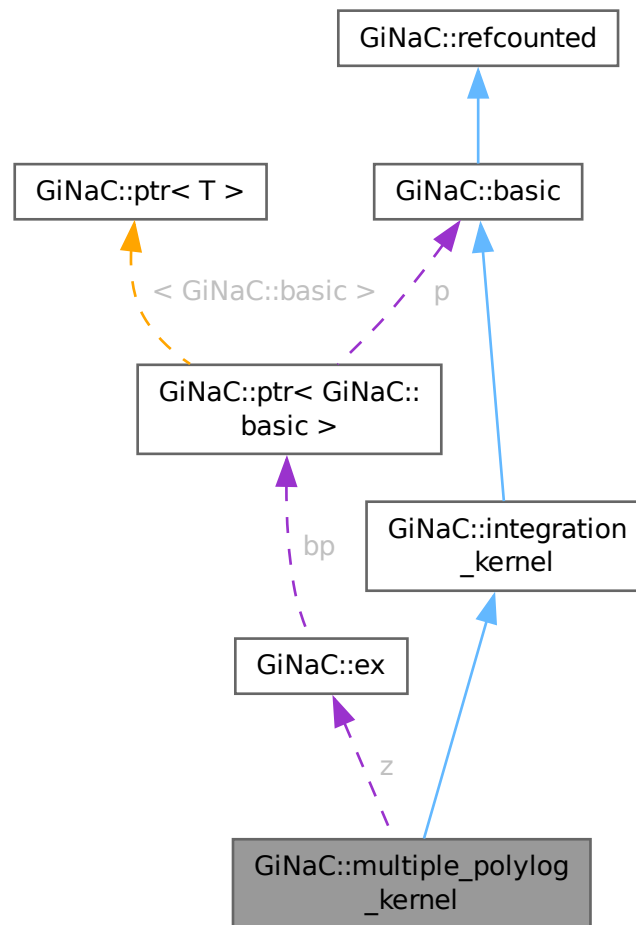
The integration kernel for multiple polylogarithms.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::multiple\_polylog\_kernel:



Collaboration diagram for GiNaC::multiple\_polylog\_kernel:



## Public Member Functions

- `multiple_polylog_kernel` (const `ex` &`z`)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `bool is_numeric (void)` const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int order, unsigned options=0) const override

- Default implementation of `ex::series()`.
- virtual bool `has_trailing_zero` (void) const

*This routine returns true, if the integration kernel has a trailing zero.*
- virtual `ex Laurent_series` (const `ex` &x, int order) const

*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const

*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const

*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const

*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const

*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N `series_coeff` (int i) const

*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()

*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const basic &other)
- const `basic` & `operator=` (const `basic` &other)

*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const

*Create a clone of this object on the heap.*
- virtual `ex eval` () const

*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const

*Evaluate object numerically.*
- virtual `ex evalm` () const

*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const

*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const

*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const

*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const



- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex](#) z

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.104.1 Detailed Description

The integration kernel for multiple polylogarithms.

This class represents the differential one-form

$$\omega^{\text{mpl}}(z) = \frac{d\lambda}{\lambda - z}$$

For the case  $z = 0$  the class [basic\\_log\\_kernel](#) should be used.

## 8.104.2 Constructor & Destructor Documentation

### 8.104.2.1 `multiple_polylog_kernel()`

```
GiNaC::multiple_polylog_kernel::multiple_polylog_kernel (
    const ex & z)
```

References [multiple\\_polylog\\_kernel\(\)](#), and [z](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), and [multiple\\_polylog\\_kernel\(\)](#).

## 8.104.3 Member Function Documentation

### 8.104.3.1 `nops()`

```
size_t GiNaC::multiple_polylog_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.104.3.2 `op()`

```
ex GiNaC::multiple_polylog_kernel::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [z](#).

### 8.104.3.3 `let_op()`

```
ex & GiNaC::multiple_polylog_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [z](#).

### 8.104.3.4 `is_numeric()`

```
bool GiNaC::multiple_polylog_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::info\\_flags::numeric](#), and [z](#).

### 8.104.3.5 series\_coeff\_impl()

```
cln::cl_N GiNaC::multiple_polylog_kernel::series_coeff_impl (
    int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex\\_to\(\)](#), and [z](#).

### 8.104.3.6 do\_print()

```
void GiNaC::multiple_polylog_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_context::s](#), and [z](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.104.4 Member Data Documentation

### 8.104.4.1 z

```
ex GiNaC::multiple_polylog_kernel::z [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [multiple\\_polylog\\_kernel\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

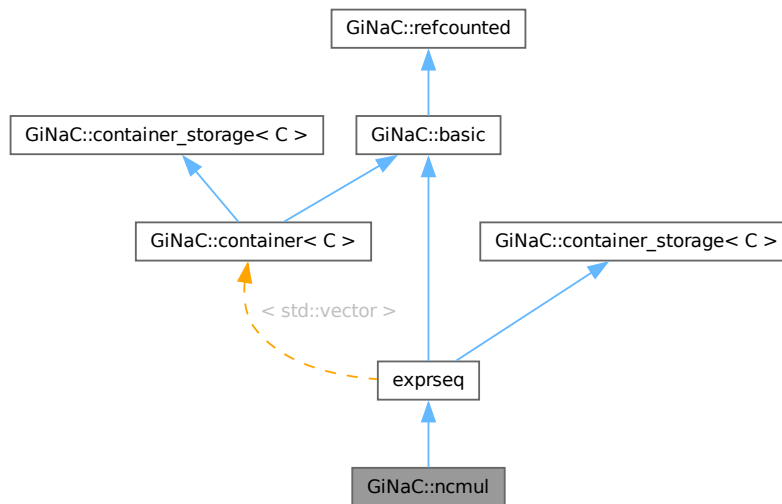
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.105 GiNaC::ncmul Class Reference

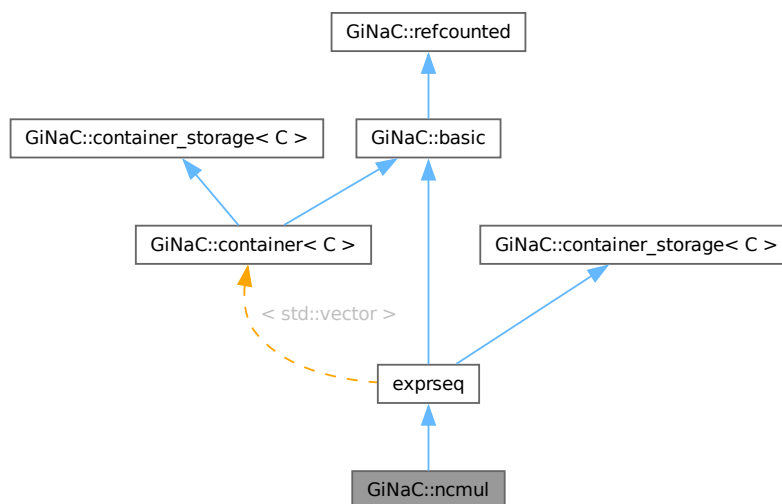
Non-commutative product of expressions.

```
#include <ncmul.h>
```

Inheritance diagram for GiNaC::ncmul:



Collaboration diagram for GiNaC::ncmul:



## Public Member Functions

- `ncmul` (const `ex` &lh, const `ex` &rh)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5, const `ex` &f6)
- `ncmul` (const `exvector` &v)
- `ncmul` (`exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- `ex coeff` (const `ex` &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex thiscontainer` (const `exvector` &v) const override  
*Similar to `duplicate()`, but with a preset sequence.*
- `ex thiscontainer` (`exvector` &&v) const override  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- const `exvector` & `get_factors` () const

Public Member Functions inherited from `GiNaC::container< std::vector >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (exvector::const\_iterator b, exvector::const\_iterator e)
- `container` (std::initializer\_list< `ex` > il)
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override

- *Load (deserialize) the object from an archive node.*
- void `archive` (`archive_node` &n) const override

*Archive the object.*

- `container` & `prepend` (const `ex` &b)

*Add element at front.*

- `container` & `append` (const `ex` &b)

*Add element at back.*

- `container` & `remove_first` ()

*Remove first element.*

- `container` & `remove_last` ()

*Remove last element.*

- `container` & `remove_all` ()

*Remove all elements.*

- `container` & `sort` ()

*Sort elements.*

- `container` & `unique` ()

*Remove adjacent duplicate elements.*

- `const_iterator` `begin` () const
- `const_iterator` `end` () const
- `const_reverse_iterator` `rbegin` () const
- `const_reverse_iterator` `rend` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex` `evalf` () const  
*Evaluate object numerically.*
- virtual `ex` `eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex` `eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex` `operator[]` (const `ex` &index) const
- virtual `ex` `operator[]` (size\_t i) const
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*



- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a non-commutative product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_csrc` (const `print_context` &c, unsigned level) const
- size\_t `count_factors` (const `ex` &e) const
- void `append_factors` (`exvector` &v, const `ex` &e) const
- `exvector expandchildren` (unsigned options) const

### Protected Member Functions inherited from `GiNaC::container< std::vector >`

- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT subchildren` (const `exmap` &m, unsigned options=0) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from `GiNaC::container_storage< C >`

- `container_storage` ()
- `container_storage` (size\_t n, const `ex` &e)
- `container_storage` (std::initializer\_list< `ex` > il)
- template<class In>  
`container_storage` (In b, In e)
- void `reserve` (size\_t)
- `~container_storage` ()
- void `reserve` (size\_t n)
- void `reserve` (std::vector< `ex` > &v, size\_t n)

**Friends**

- class [power](#)
- [ex reeval\\_ncmul](#) (const [exvector](#) &v)
- [ex hold\\_ncmul](#) (const [exvector](#) &v)

**Additional Inherited Members****Public Types inherited from [GiNaC::container](#)< [std::vector](#) >**

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

**Protected Types inherited from [GiNaC::container](#)< [std::vector](#) >**

- typedef [container\\_storage](#)< [std::vector](#) >::STLT STLT

**Protected Types inherited from [GiNaC::container\\_storage](#)< [C](#) >**

- typedef [C](#)< [ex](#) > STLT

**Static Protected Member Functions inherited from [GiNaC::container](#)< [std::vector](#) >**

- static unsigned [get\\_default\\_flags](#) ()
- static char [get\\_open\\_delim](#) ()
- static char [get\\_close\\_delim](#) ()

**Static Protected Member Functions inherited from [GiNaC::container\\_storage](#)< [C](#) >**

- static void [reserve](#) (STLT &, size\_t)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

**Protected Attributes inherited from [GiNaC::container\\_storage](#)< [C](#) >**

- [STLT seq](#)

**8.105.1 Detailed Description**

Non-commutative product of expressions.

## 8.105.2 Constructor & Destructor Documentation

### 8.105.2.1 ncmul() [1/7]

```
GiNaC::ncmul::ncmul (  
    const ex & lh,  
    const ex & rh)
```

Referenced by [power](#).

### 8.105.2.2 ncmul() [2/7]

```
GiNaC::ncmul::ncmul (  
    const ex & f1,  
    const ex & f2,  
    const ex & f3)
```

### 8.105.2.3 ncmul() [3/7]

```
GiNaC::ncmul::ncmul (  
    const ex & f1,  
    const ex & f2,  
    const ex & f3,  
    const ex & f4)
```

### 8.105.2.4 ncmul() [4/7]

```
GiNaC::ncmul::ncmul (  
    const ex & f1,  
    const ex & f2,  
    const ex & f3,  
    const ex & f4,  
    const ex & f5)
```

### 8.105.2.5 ncmul() [5/7]

```
GiNaC::ncmul::ncmul (  
    const ex & f1,  
    const ex & f2,  
    const ex & f3,  
    const ex & f4,  
    const ex & f5,  
    const ex & f6)
```

### 8.105.2.6 ncmul() [6/7]

```
GiNaC::ncmul::ncmul (  
    const exvector & v)
```

### 8.105.2.7 ncmul() [7/7]

```
GiNaC::ncmul::ncmul (
    exvector && v)
```

## 8.105.3 Member Function Documentation

### 8.105.3.1 precedence()

```
unsigned GiNaC::ncmul::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< std::vector >](#).

References [precedence\(\)](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), and [precedence\(\)](#).

### 8.105.3.2 info()

```
bool GiNaC::ncmul::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< std::vector >](#).

### 8.105.3.3 degree()

```
int GiNaC::ncmul::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex\\_to\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

### 8.105.3.4 ldegree()

```
int GiNaC::ncmul::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex\\_to\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

### 8.105.3.5 expand()

```
ex GiNaC::ncmul::expand (
    unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::dynallocate\(\)](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::container< std::vector >::op\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

### 8.105.3.6 coeff()

```
ex GiNaC::ncmul::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::add::coeff\(\)](#).

### 8.105.3.7 eval()

```
ex GiNaC::ncmul::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- $\text{ncmul}(\dots, *(x1, x2), \dots, \text{ncmul}(x3, x4), \dots) \rightarrow \text{ncmul}(\dots, x1, x2, \dots, x3, x4, \dots)$  (associativity)
- $\text{ncmul}(x) \rightarrow x$
- $\text{ncmul}() \rightarrow 1$
- $\text{ncmul}(\dots, c1, \dots, c2, \dots) \rightarrow *(c1, c2, \text{ncmul}(\dots))$  (pull out commutative elements)
- $\text{ncmul}(x1, y1, x2, y2) \rightarrow *(\text{ncmul}(x1, x2), \text{ncmul}(y1, y2))$  (collect elements of same type)
- $\text{ncmul}(x1, x2, x3, \dots) \rightarrow x::\text{eval\_ncmul}(x1, x2, x3, \dots)$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::factor\(\)](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.105.3.8 evalm()**

```
ex GiNaC::ncmul::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::matrix::mul\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**8.105.3.9 get\_free\_indices()**

```
exvector GiNaC::ncmul::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::container< std::vector >::nops\(\)](#), and [GiNaC::container< std::vector >::op\(\)](#).

**8.105.3.10 thiscontainer() [1/2]**

```
ex GiNaC::ncmul::thiscontainer (
    const exvector & v) const [override], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::dynallocate\(\)](#).

**8.105.3.11 thiscontainer() [2/2]**

```
ex GiNaC::ncmul::thiscontainer (
    exvector && v) const [override], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::dynallocate\(\)](#).

**8.105.3.12 conjugate()**

```
ex GiNaC::ncmul::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::container< std::vector >::begin\(\)](#), [GiNaC::container< std::vector >::conjugate\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::container< std::vector >::end\(\)](#), [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::container< std::vector >::nops\(\)](#), [return\\_type\(\)](#), and [return\\_type\\_tinfo\(\)](#).

**8.105.3.13 real\_part()**

```
ex GiNaC::ncmul::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::basic::real\\_part\(\)](#).

**8.105.3.14 imag\_part()**

```
ex GiNaC::ncmul::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< std::vector >](#).

References [GiNaC::basic::imag\\_part\(\)](#).

**8.105.3.15 derivative()**

```
ex GiNaC::ncmul::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a non-commutative product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::ex::swap\(\)](#).

**8.105.3.16 return\_type()**

```
unsigned GiNaC::ncmul::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< std::vector >::end\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).



**8.105.3.17 return\_type\_tinfo()**

```
return_type_t GiNaC::ncmul::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::make\\_return\\_type\\_t\(\)](#), [GiNaC::return\\_types::noncommutative](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**8.105.3.18 do\_print()**

```
void GiNaC::ncmul::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [precedence\(\)](#), and [GiNaC::container< std::vector >::printseq\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.105.3.19 do\_print\_csrc()**

```
void GiNaC::ncmul::do_print_csrc (
    const print_context & c,
    unsigned level) const [protected]
```

References [precedence\(\)](#), [GiNaC::container< std::vector >::printseq\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.105.3.20 count\_factors()**

```
size_t GiNaC::ncmul::count_factors (
    const ex & e) const [protected]
```

References [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [count\\_factors\(\)](#), and [eval\(\)](#).

**8.105.3.21 append\_factors()**

```
void GiNaC::ncmul::append_factors (
    exvector & v,
    const ex & e) const [protected]
```

References [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [append\\_factors\(\)](#), and [eval\(\)](#).

**8.105.3.22 expandchildren()**

```
exvector GiNaC::ncmul::expandchildren (
    unsigned options) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container< std::vector >::end\(\)](#), [GiNaC::ex::expand\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [expand\(\)](#).

**8.105.3.23 get\_factors()**

```
const exvector & GiNaC::ncmul::get_factors () const
```

References [GiNaC::container\\_storage< C >::seq](#).

**8.105.4 Friends And Related Symbol Documentation****8.105.4.1 power**

```
friend class power [friend]
```

References [hold\\_ncmul](#), [ncmul\(\)](#), [power](#), and [reeval\\_ncmul](#).

Referenced by [power](#).

**8.105.4.2 reeval\_ncmul**

```
ex reeval_ncmul (
    const exvector & v) [friend]
```

References [GiNaC::dynallocate\(\)](#).

Referenced by [power](#).

**8.105.4.3 hold\_ncmul**

```
ex hold_ncmul (
    const exvector & v) [friend]
```

References [GiNaC::\\_ex1](#), [GiNaC::dynallocate\(\)](#), and [GiNaC::status\\_flags::evaluated](#).

Referenced by [power](#).

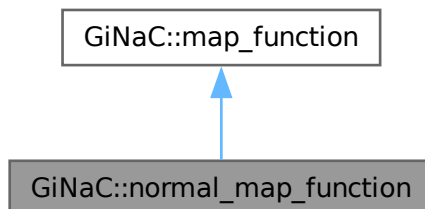
The documentation for this class was generated from the following files:

- [ncmul.h](#)
- [indexed.cpp](#)
- [ncmul.cpp](#)

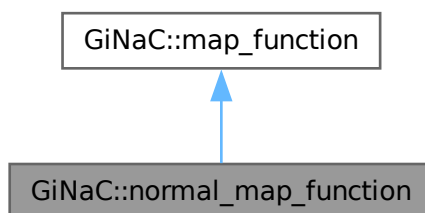
## 8.106 GiNaC::normal\_map\_function Struct Reference

Function object to be applied by [basic::normal\(\)](#).

Inheritance diagram for GiNaC::normal\_map\_function:



Collaboration diagram for GiNaC::normal\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.106.1 Detailed Description

Function object to be applied by [basic::normal\(\)](#).

### 8.106.2 Member Function Documentation

#### 8.106.2.1 operator>()

```
ex GiNaC::normal_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::normal\(\)](#).

The documentation for this struct was generated from the following file:

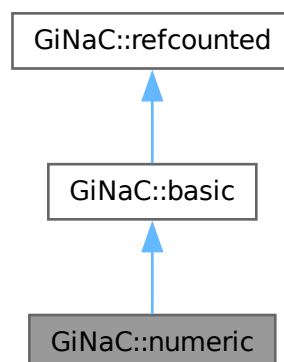
- [normal.cpp](#)

## 8.107 GiNaC::numeric Class Reference

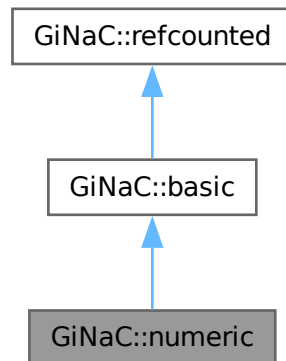
This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::numeric:



Collaboration diagram for GiNaC::numeric:



## Public Member Functions

- [numeric](#) (int i)
- [numeric](#) (unsigned int i)
- [numeric](#) (long i)
- [numeric](#) (unsigned long i)
- [numeric](#) (long long i)
- [numeric](#) (unsigned long long i)
- [numeric](#) (long [numer](#), long [denom](#))  
*Constructor for rational numerics a/b.*
- [numeric](#) (double d)
- [numeric](#) (const char \*)  
*ctor from C-style string.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- bool [is\\_polynomial](#) (const [ex](#) &s) const override  
*Check whether this is a polynomial in the given variables.*
- int [degree](#) (const [ex](#) &s) const override  
*Return degree of highest power in object s.*
- int [ldegree](#) (const [ex](#) &s) const override  
*Return degree of lowest power in object s.*
- [ex](#) [coeff](#) (const [ex](#) &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- bool [has](#) (const [ex](#) &other, unsigned options=0) const override  
*Disassemble real part and imaginary part to scan for the occurrence of a single number.*
- [ex](#) [eval](#) () const override  
*Evaluation of numbers doesn't do anything at all.*
- [ex](#) [evalf](#) () const override  
*Cast numeric into a floating-point object.*

- `ex subs` (const `exmap` &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Implementation of `ex::normal()` for a numeric.*
- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for a numeric.*
- `ex to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for a numeric.*
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- const `numeric add` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric sub` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric mul` (const `numeric` &other) const  
*Numerical multiplication method.*
- const `numeric div` (const `numeric` &other) const  
*Numerical division method.*
- const `numeric power` (const `numeric` &other) const  
*Numerical exponentiation.*
- const `numeric & add_dyn` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric & sub_dyn` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric & mul_dyn` (const `numeric` &other) const  
*Numerical multiplication method.*
- const `numeric & div_dyn` (const `numeric` &other) const  
*Numerical division method.*
- const `numeric & power_dyn` (const `numeric` &other) const  
*Numerical exponentiation.*
- const `numeric & operator=` (int i)
- const `numeric & operator=` (unsigned int i)
- const `numeric & operator=` (long i)
- const `numeric & operator=` (unsigned long i)
- const `numeric & operator=` (double d)
- const `numeric & operator=` (const char \*s)
- const `numeric inverse` () const  
*Inverse of a number.*
- `numeric step` () const  
*Return the step function of a numeric.*
- int `csgn` () const

- Return the complex half-plane (left or right) in which the number lies.*

  - int `compare` (const `numeric` &other) const

*This method establishes a canonical order on all numbers.*
- bool `is_equal` (const `numeric` &other) const
- bool `is_zero` () const
- True if object is zero.*

  - bool `is_positive` () const

*True if object is not complex and greater than zero.*
- bool `is_negative` () const
- True if object is not complex and less than zero.*

  - bool `is_integer` () const

*True if object is a non-complex integer.*
- bool `is_pos_integer` () const
- True if object is an exact integer greater than zero.*

  - bool `is_nonneg_integer` () const

*True if object is an exact integer greater or equal zero.*
- bool `is_even` () const
- True if object is an exact even integer.*

  - bool `is_odd` () const

*True if object is an exact odd integer.*
- bool `is_prime` () const
- Probabilistic primality test.*

  - bool `is_rational` () const

*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool `is_real` () const
- True if object is a real integer, rational or float (but not complex).*

  - bool `is_cinteger` () const

*True if object is element of the domain of integers extended by I, i.e.*
- bool `is_crational` () const
- True if object is an exact rational number, may even be complex (denominator may be unity).*

  - bool `operator==` (const `numeric` &other) const
  - bool `operator!=` (const `numeric` &other) const
  - bool `operator<` (const `numeric` &other) const

*Numerical comparison: less.*
- bool `operator<=` (const `numeric` &other) const
- Numerical comparison: less or equal.*

  - bool `operator>` (const `numeric` &other) const

*Numerical comparison: greater.*
- bool `operator>=` (const `numeric` &other) const
- Numerical comparison: greater or equal.*

  - int `to_int` () const

*Converts numeric types to machine's int.*
- long `to_long` () const
- Converts numeric types to machine's long.*

  - double `to_double` () const

*Converts numeric types to machine's double.*
- `cln::cl_N` `to_cl_N` () const
- Returns a new CLN object of type `cl_N`, representing the value of `*this`.*

  - const `numeric real` () const

*Real part of a number.*
- const `numeric imag` () const

- *Imaginary part of a number.*
- const `numeric numer` () const  
*Numerator.*
- const `numeric denom` () const  
*Denominator.*
- int `int_length` () const  
*Size in binary notation.*
- `numeric` (const `cln::cl_N` &z)  
*Ctor from CLN types.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*



- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff` for a numeric always returns 0.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void [print\\_numeric](#) (const [print\\_context](#) &c, const char \*par\_open, const char \*par\_close, const char \*imag↔\_sym, const char \*mul\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_csrc\\_cl\\_N](#) (const [print\\_csrc\\_cl\\_N](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [cln::cl\\_N](#) [value](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.107.1 Detailed Description

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

Objects of this type may directly be created by the user.

## 8.107.2 Constructor & Destructor Documentation

### 8.107.2.1 numeric() [1/10]

```
GiNaC::numeric::numeric (
    int i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

Referenced by [add\(\)](#), [add\\_dyn\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [denom\(\)](#), [div\(\)](#), [div\\_dyn\(\)](#), [evalf\(\)](#), [has\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [integer\\_content\(\)](#), [inverse\(\)](#), [is\\_equal\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [max\\_coefficient\(\)](#), [mul\(\)](#), [mul\\_dyn\(\)](#), [normal\(\)](#), [numer\(\)](#), [operator!=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [power\(\)](#), [power\\_dyn\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [smod\(\)](#), [step\(\)](#), [sub\(\)](#), [sub\\_dyn\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 8.107.2.2 numeric() [2/10]

```
GiNaC::numeric::numeric (
    unsigned int i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 8.107.2.3 numeric() [3/10]

```
GiNaC::numeric::numeric (
    long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 8.107.2.4 numeric() [4/10]

```
GiNaC::numeric::numeric (
    unsigned long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 8.107.2.5 numeric() [5/10]

```
GiNaC::numeric::numeric (
    long long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 8.107.2.6 numeric() [6/10]

```
GiNaC::numeric::numeric (
    unsigned long long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.2.7 numeric() [7/10]**

```
GiNaC::numeric::numeric (
    long numer,
    long denom)
```

Constructor for rational numerics a/b.

**Exceptions**

<code>overflow_error</code>	(division by zero)
-----------------------------	--------------------

References [denom\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [numer\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.2.8 numeric() [8/10]**

```
GiNaC::numeric::numeric (
    double d)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.2.9 numeric() [9/10]**

```
GiNaC::numeric::numeric (
    const char * s)
```

ctor from C-style string.

It also accepts complex numbers in [GiNaC](#) notation like "2+5\*I".

References [GiNaC::Digits](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.2.10 numeric() [10/10]**

```
GiNaC::numeric::numeric (
    const cln::cl_N & z) [explicit]
```

Ctor from CLN types.

This is for the initiated user or internal use only.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.3 Member Function Documentation****8.107.3.1 precedence()**

```
unsigned GiNaC::numeric::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_numeric\(\)](#).

### 8.107.3.2 info()

```
bool GiNaC::numeric::info (
    unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [GiNaC::zeta1\\_eval\(\)](#).

### 8.107.3.3 is\_polynomial()

```
bool GiNaC::numeric::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.107.3.4 degree()

```
int GiNaC::numeric::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.107.3.5 ldegree()

```
int GiNaC::numeric::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.107.3.6 `coeff()`

```
ex GiNaC::numeric::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and [GiNaC::basic::ex](#).

Referenced by [GiNaC::pseries::mul\\_const\(\)](#).

### 8.107.3.7 `has()`

```
bool GiNaC::numeric::has (
    const ex & other,
    unsigned options = 0) const [override], [virtual]
```

Disassemble real part and imaginary part to scan for the occurrence of a single number.

Also handles the imaginary unit. It ignores the sign on both this and the argument, which may lead to what might appear as funny results:  $(2+i).has(-2) \rightarrow true$ . But this is consistent, since we also would like to have  $(-2+i).has(2) \rightarrow true$  and we want to think about the sign as a multiplicative factor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::l](#), [imag\(\)](#), [is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [numeric\(\)](#), and [real\(\)](#).

### 8.107.3.8 `eval()`

```
ex GiNaC::numeric::eval () const [override], [virtual]
```

Evaluation of numbers doesn't do anything at all.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::basic::hold\(\)](#).

### 8.107.3.9 `evalf()`

```
ex GiNaC::numeric::evalf () const [override], [virtual]
```

Cast numeric into a floating-point object.

For example `exact numeric(1)` is returned as a 1.00000000000000000000 and so on according to how Digits is currently set. In case the object already was a floating point number the precision is trimmed to match the currently set default.

Returns

an ex-handle to a numeric.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [numeric\(\)](#), and [value](#).

### 8.107.3.10 subs()

```
ex GiNaC::numeric::subs (
    const exmap & m,
    unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 8.107.3.11 normal()

```
ex GiNaC::numeric::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [denom\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [numer\(\)](#), [numeric\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

### 8.107.3.12 to\_rational()

```
ex GiNaC::numeric::to_rational (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::I](#), [imag\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [numeric\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**8.107.3.13 to\_polynomial()**

```
ex GiNaC::numeric::to_polynomial (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-integer real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_real\(\)](#), [numeric\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**8.107.3.14 integer\_content()**

```
numeric GiNaC::numeric::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), and [numeric\(\)](#).

Referenced by [GiNaC::mul::eval\(\)](#).

**8.107.3.15 smod()**

```
ex GiNaC::numeric::smod (
    const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$xi$	modulus
------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [numeric\(\)](#), and [GiNaC::smod\(\)](#).



### 8.107.3.16 max\_coefficient()

```
numeric GiNaC::numeric::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), and [numeric\(\)](#).

### 8.107.3.17 conjugate()

```
ex GiNaC::numeric::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

### 8.107.3.18 real\_part()

```
ex GiNaC::numeric::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [numeric\(\)](#), and [value](#).

### 8.107.3.19 imag\_part()

```
ex GiNaC::numeric::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [numeric\(\)](#), and [value](#).

### 8.107.3.20 archive()

```
void GiNaC::numeric::archive (  
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_string\(\)](#), [value](#), and [GiNaC::write\\_real\\_float\(\)](#).

**8.107.3.21 read\_archive()**

```
void GiNaC::numeric::read_archive (
    const archive\_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::read\\_real\\_float\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.3.22 derivative()**

```
ex GiNaC::numeric::derivative (
    const symbol & s) const [inline], [override], [protected], [virtual]
```

Implementation of [ex::diff](#) for a numeric always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.107.3.23 is\_equal\_same\_type()**

```
bool GiNaC::numeric::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [GINAC\\_ASSERT](#), [is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [numeric\(\)](#).

**8.107.3.24 calchash()**

```
unsigned GiNaC::numeric::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**8.107.3.25 add()**

```
const numeric GiNaC::numeric::add (
    const numeric & other) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator+\(\)](#), [GiNaC::operator++\(\)](#), [GiNaC::operator++\(\)](#), [GiNaC::operator+=\(\)](#), [GiNaC::operator--\(\)](#), and [GiNaC::operator--\(\)](#).

**8.107.3.26 sub()**

```
const numeric GiNaC::numeric::sub (
    const numeric & other) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::operator-\(\)](#), [GiNaC::operator-=\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

**8.107.3.27 mul()**

```
const numeric GiNaC::numeric::mul (
    const numeric & other) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator\\*\(\)](#), [GiNaC::operator\\*=\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

**8.107.3.28 div()**

```
const numeric GiNaC::numeric::div (
    const numeric & other) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object.

**Exceptions**

<a href="#">overflow_error</a>	(division by zero)
--------------------------------	--------------------

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator/\(\)](#), [GiNaC::operator/\(\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

### 8.107.3.29 power()

```
const numeric GiNaC::numeric::power (
    const numeric & other) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), and [GiNaC::pow\(\)](#).

### 8.107.3.30 add\_dyn()

```
const numeric & GiNaC::numeric::add_dyn (
    const numeric & other) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), [GiNaC::dynallocate\(\)](#), [numeric\(\)](#), and [value](#).

### 8.107.3.31 sub\_dyn()

```
const numeric & GiNaC::numeric::sub_dyn (
    const numeric & other) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), [GiNaC::dynallocate\(\)](#), [numeric\(\)](#), and [value](#).

### 8.107.3.32 mul\_dyn()

```
const numeric & GiNaC::numeric::mul_dyn (
    const numeric & other) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num1\\_p](#), [GiNaC::dynallocate\(\)](#), [numeric\(\)](#), and [value](#).

**8.107.3.33 div\_dyn()**

```
const numeric & GiNaC::numeric::div_dyn (
    const numeric & other) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

**Exceptions**

<i>overflow_error</i>	(division by zero)
-----------------------	--------------------

References [GiNaC::\\_num1\\_p](#), [GiNaC::dynallocate\(\)](#), [numeric\(\)](#), and [value](#).

**8.107.3.34 power\_dyn()**

```
const numeric & GiNaC::numeric::power_dyn (
    const numeric & other) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::dynallocate\(\)](#), [numeric\(\)](#), and [value](#).

**8.107.3.35 operator=() [1/6]**

```
const numeric & GiNaC::numeric::operator= (
    int i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**8.107.3.36 operator=() [2/6]**

```
const numeric & GiNaC::numeric::operator= (
    unsigned int i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**8.107.3.37 operator=() [3/6]**

```
const numeric & GiNaC::numeric::operator= (
    long i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**8.107.3.38 operator=()** [4/6]

```
const numeric & GiNaC::numeric::operator= (  
    unsigned long i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**8.107.3.39 operator=()** [5/6]

```
const numeric & GiNaC::numeric::operator= (  
    double d)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**8.107.3.40 operator=()** [6/6]

```
const numeric & GiNaC::numeric::operator= (  
    const char * s)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**8.107.3.41 inverse()**

```
const numeric GiNaC::numeric::inverse () const
```

Inverse of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::inverse\(\)](#), and [GiNaC::psi1\\_eval\(\)](#).

**8.107.3.42 step()**

```
numeric GiNaC::numeric::step () const
```

Return the step function of a numeric.

The imaginary part of it is ignored because the step function is generally considered real but a numeric may develop a small imaginary part due to rounding errors.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::step\(\)](#).

**8.107.3.43 csgn()**

```
int GiNaC::numeric::csgn () const
```

Return the complex half-plane (left or right) in which the number lies.

$\text{csgn}(x) == 0$  for  $x == 0$ ,  $\text{csgn}(x) == 1$  for  $\text{Re}(x) > 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) > 0$ ,  $\text{csgn}(x) == -1$  for  $\text{Re}(x) < 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) < 0$ .

See also

`numeric::compare(const numeric &other)`

References [value](#).

Referenced by [GiNaC::csgn\(\)](#).

**8.107.3.44 compare()**

```
int GiNaC::numeric::compare (
    const numeric & other) const
```

This method establishes a canonical order on all numbers.

For complex numbers this is not possible in a mathematically consistent way but we need to establish some order and it ought to be fast. So we simply define it to be compatible with our method `csgn`.

Returns

`csgn(*this-other)`

See also

[numeric::csgn\(\)](#)

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::binomial\(\)](#), and [GiNaC::power::eval\(\)](#).

**8.107.3.45 is\_equal()**

```
bool GiNaC::numeric::is_equal (
    const numeric & other) const
```

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::cos\\_eval\(\)](#), [GiNaC::doublefactorial\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [has\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::tan\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

**8.107.3.46 is\_zero()**

```
bool GiNaC::numeric::is_zero () const
```

True if object is zero.

References [value](#).

Referenced by [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::clifford\\_to\\_1st\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_expairseq\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [has\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::log\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::step\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

**8.107.3.47 is\_positive()**

```
bool GiNaC::numeric::is_positive () const
```

True if object is not complex and greater than zero.

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [info\(\)](#), [GiNaC::is\\_positive\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::Gt::qExpand\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

**8.107.3.48 is\_negative()**

```
bool GiNaC::numeric::is_negative () const
```

True if object is not complex and less than zero.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fibonacci\(\)](#), [info\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), and [GiNaC::Gt::qExpand\(\)](#).

**8.107.3.49 is\_integer()**

```
bool GiNaC::numeric::is_integer () const
```

True if object is a non-complex integer.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::Gt::decomposeIntegration](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::isqrt\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::mod\(\)](#), [normal\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [to\\_int\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), [to\\_long\(\)](#), [to\\_polynomial\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).



### 8.107.3.50 is\_pos\_integer()

```
bool GiNaC::numeric::is_pos_integer () const
```

True if object is an exact integer greater than zero.

References [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::ifactor\(\)](#), [info\(\)](#), [GiNaC::is\\_pos\\_integer\(\)](#), and [GiNaC::pseries::power\\_const\(\)](#).

### 8.107.3.51 is\_nonneg\_integer()

```
bool GiNaC::numeric::is_nonneg_integer () const
```

True if object is an exact integer greater or equal zero.

References [value](#).

Referenced by [GiNaC::binomial\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::doublefactorial\(\)](#), [GiNaC::factorial\(\)](#), [info\(\)](#), [GiNaC::is\\_nonneg\\_integer\(\)](#), [print\\_numeric\(\)](#), and [GiNaC::Gt::to\\_integer\(\)](#).

### 8.107.3.52 is\_even()

```
bool GiNaC::numeric::is_even () const
```

True if object is an exact even integer.

References [value](#).

Referenced by [GiNaC::fibonacci\(\)](#), [info\(\)](#), [GiNaC::is\\_even\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

### 8.107.3.53 is\_odd()

```
bool GiNaC::numeric::is_odd () const
```

True if object is an exact odd integer.

References [value](#).

Referenced by [info\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::is\\_odd\(\)](#), and [GiNaC::matrix::pow\(\)](#).

### 8.107.3.54 is\_prime()

```
bool GiNaC::numeric::is_prime () const
```

Probabilistic primality test.

Returns

true if object is exact integer and prime.

References [value](#).

Referenced by [info\(\)](#), and [GiNaC::is\\_prime\(\)](#).

**8.107.3.55 is\_rational()**

```
bool GiNaC::numeric::is_rational () const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), [GiNaC::is\\_rational\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [normal\(\)](#), and [to\\_rational\(\)](#).

**8.107.3.56 is\_real()**

```
bool GiNaC::numeric::is_real () const
```

True if object is a real integer, rational or float (but not complex).

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::beta\\_eval\(\)](#), [conjugate\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [denom\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fsolve\(\)](#), [has\(\)](#), [info\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [GiNaC::is\\_real\(\)](#), [normal\(\)](#), [numer\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_double\(\)](#), [to\\_polynomial\(\)](#), [to\\_rational\(\)](#), and [GiNaC::zeta\(\)](#).

**8.107.3.57 is\_cinteger()**

```
bool GiNaC::numeric::is_cinteger () const
```

True if object is element of the domain of integers extended by  $i$ , i.e.

is of the form  $a+bi$ , where  $a$  and  $b$  are integers.

References [is\\_real\(\)](#), and [value](#).

Referenced by [info\(\)](#), and [GiNaC::is\\_cinteger\(\)](#).

**8.107.3.58 is\_crational()**

```
bool GiNaC::numeric::is_crational () const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [is\\_real\(\)](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), and [GiNaC::is\\_crational\(\)](#).

**8.107.3.59 operator==(**

```
bool GiNaC::numeric::operator== (
    const numeric & other) const
```

References [numeric\(\)](#), and [value](#).

### 8.107.3.60 operator"!=()

```
bool GiNaC::numeric::operator!= (
    const numeric & other) const
```

References [numeric\(\)](#), and [value](#).

### 8.107.3.61 operator<()

```
bool GiNaC::numeric::operator< (
    const numeric & other) const
```

Numerical comparison: less.

#### Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

### 8.107.3.62 operator<=()

```
bool GiNaC::numeric::operator<= (
    const numeric & other) const
```

Numerical comparison: less or equal.

#### Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

### 8.107.3.63 operator>()

```
bool GiNaC::numeric::operator> (
    const numeric & other) const
```

Numerical comparison: greater.

#### Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

**8.107.3.64 operator>=()**

```
bool GiNaC::numeric::operator>= (
    const numeric & other) const
```

Numerical comparison: greater or equal.

**Exceptions**

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

**8.107.3.65 to\_int()**

```
int GiNaC::numeric::to_int () const
```

Converts numeric types to machine's int.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::doublefactorial\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::to\\_int\(\)](#).

**8.107.3.66 to\_long()**

```
long GiNaC::numeric::to_long () const
```

Converts numeric types to machine's long.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::Gt::to\\_integer\(\)](#), and [GiNaC::to\\_long\(\)](#).

**8.107.3.67 to\_double()**

```
double GiNaC::numeric::to_double () const
```

Converts numeric types to machine's double.

You should check with [is\\_real\(\)](#) if the number is really not complex before calling this method.

References [GINAC\\_ASSERT](#), [is\\_real\(\)](#), and [value](#).

Referenced by [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::Gt::qExpand\(\)](#), and [GiNaC::to\\_double\(\)](#).

**8.107.3.68 to\_cl\_N()**

```
cln::cl_N GiNaC::numeric::to_cl_N () const
```

Returns a new CLN object of type `cl_N`, representing the value of `*this`.

This method may be used when mixing [GiNaC](#) and CLN in one project.

References [value](#).

Referenced by [GiNaC::abs\(\)](#), [GiNaC::acos\(\)](#), [GiNaC::acosh\(\)](#), [GiNaC::asin\(\)](#), [GiNaC::asinh\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atanh\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::cosh\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::lgamma\(\)](#), [GiNaC::Li2\(\)](#), [GiNaC::log\(\)](#), [GiNaC::mod\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::sin\(\)](#), [GiNaC::sinh\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::sqrt\(\)](#), [GiNaC::tan\(\)](#), [GiNaC::tanh\(\)](#), [GiNaC::tgamma\(\)](#), and [GiNaC::zeta\(\)](#).

**8.107.3.69 real()**

```
const numeric GiNaC::numeric::real () const
```

Real part of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::power::eval\(\)](#), [has\(\)](#), [normal\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::real\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), [to\\_rational\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

**8.107.3.70 imag()**

```
const numeric GiNaC::numeric::imag () const
```

Imaginary part of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::Gt::findMoebiusTransform\(\)](#), [has\(\)](#), [GiNaC::imag\(\)](#), [normal\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), [to\\_rational\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

**8.107.3.71 numer()**

```
const numeric GiNaC::numeric::numer () const
```

Numerator.

Computes the numerator of rational numbers, rationalized numerator of complex if real and imaginary part are both rational numbers (i.e `numer(4/3+5/6*I) == 8+5*I`), the number carrying the sign in all other cases.

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), [GiNaC::numer\(\)](#), and [numeric\(\)](#).

**8.107.3.72 `denom()`**

```
const numeric GiNaC::numeric::denom () const
```

Denominator.

Computes the denominator of rational numbers, common integer denominator of complex if real and imaginary part are both rational numbers (i.e `denom(4/3+5/6*I) == 6`), one in all other cases.

References [GiNaC::\\_num1\\_p](#), [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::denom\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), and [numeric\(\)](#).

**8.107.3.73 `int_length()`**

```
int GiNaC::numeric::int_length () const
```

Size in binary notation.

For integers, this is the smallest  $n \geq 0$  such that  $-2^n \leq x < 2^n$ . If  $x > 0$ , this is the unique  $n > 0$  such that  $2^{n-1} \leq x < 2^n$ .

**Returns**

number of bits (excluding sign) needed to represent that number in two's complement if it is an integer, 0 otherwise.

References [value](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

**8.107.3.74 `print_numeric()`**

```
void GiNaC::numeric::print_numeric (
    const print_context & c,
    const char * par_open,
    const char * par_close,
    const char * imag_sym,
    const char * mul_sym,
    unsigned level) const [protected]
```

References [is\\_nonneg\\_integer\(\)](#), [precedence\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

**8.107.3.75 `do_print()`**

```
void GiNaC::numeric::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [print\\_numeric\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.107.3.76 do\_print\_latex()**

```
void GiNaC::numeric::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [print\\_numeric\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.107.3.77 do\_print\_csrc()**

```
void GiNaC::numeric::do_print_csrc (
    const print\_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::is\\_a\(\)](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.107.3.78 do\_print\_csrc\_cl\_N()**

```
void GiNaC::numeric::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level) const [protected]
```

References [is\\_real\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.107.3.79 do\_print\_tree()**

```
void GiNaC::numeric::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.107.3.80 do\_print\_python\_repr()**

```
void GiNaC::numeric::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [print\\_numeric\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.107.4 Member Data Documentation

### 8.107.4.1 value

`cln::cl_N GiNaC::numeric::value` [protected]

Referenced by [add\(\)](#), [add\\_dyn\(\)](#), [archive\(\)](#), [calchash\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [csgn\(\)](#), [denom\(\)](#), [div\(\)](#), [div\\_dyn\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_tree\(\)](#), [evalf\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [int\\_length\(\)](#), [inverse\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_equal\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [mul\(\)](#), [mul\\_dyn\(\)](#), [numer\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [operator!=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [power\(\)](#), [power\\_dyn\(\)](#), [print\\_numeric\(\)](#), [read\\_archive\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [step\(\)](#), [sub\(\)](#), [sub\\_dyn\(\)](#), [to\\_cl\\_N\(\)](#), [to\\_double\(\)](#), [to\\_int\(\)](#), and [to\\_long\(\)](#).

The documentation for this class was generated from the following files:

- [numeric.h](#)
- [normal.cpp](#)
- [numeric.cpp](#)

## 8.108 GiNaC::op0\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- `bool operator\(\) (const ex &lh, const ex &rh) const`

## 8.108.1 Member Function Documentation

### 8.108.1.1 [operator\(\)](#)()

```
bool GiNaC::op0_is_equal::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

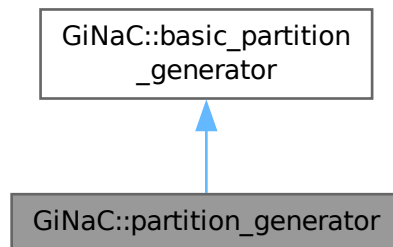


## 8.109 GiNaC::partition\_generator Class Reference

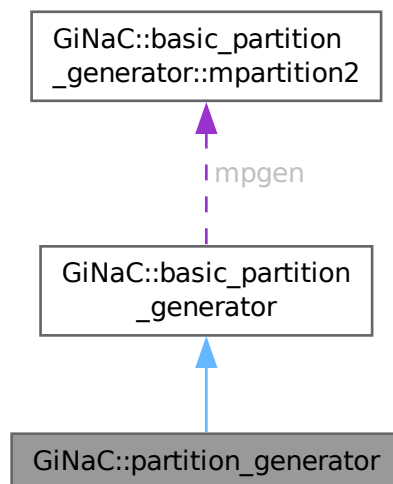
Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::partition\_generator:



Collaboration diagram for GiNaC::partition\_generator:



### Public Member Functions

- [partition\\_generator](#) (unsigned  $n$ , unsigned  $m$ )
- `const std::vector< unsigned > & get () const`
- `bool next ()`

**Private Attributes**

- `std::vector< unsigned > partition`
- `bool current_updated`

**Additional Inherited Members****Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)**

- [basic\\_partition\\_generator](#) (unsigned n\_, unsigned m\_)

**Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)**

- `mpartition2 mpgen`

**8.109.1 Detailed Description**

Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order.

**8.109.2 Constructor & Destructor Documentation****8.109.2.1 partition\_generator()**

```
GiNaC::partition_generator::partition_generator (
    unsigned n_,
    unsigned m_) [inline]
```

References [GiNaC::basic\\_partition\\_generator::basic\\_partition\\_generator\(\)](#), [current\\_updated](#), and [partition](#).

**8.109.3 Member Function Documentation****8.109.3.1 get()**

```
const std::vector< unsigned > & GiNaC::partition_generator::get () const [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), and [partition](#).

**8.109.3.2 next()**

```
bool GiNaC::partition_generator::next () [inline]
```

References [current\\_updated](#), and [GiNaC::basic\\_partition\\_generator::mpgen](#).

## 8.109.4 Member Data Documentation

### 8.109.4.1 partition

```
std::vector<unsigned> GiNaC::partition_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#), and [partition\\_generator\(\)](#).

### 8.109.4.2 current\_updated

```
bool GiNaC::partition_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), [next\(\)](#), and [partition\\_generator\(\)](#).

The documentation for this class was generated from the following file:

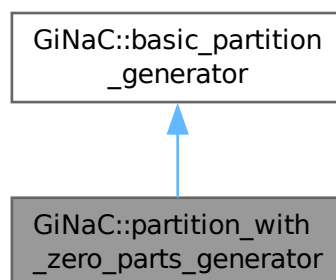
- [utils.h](#)

## 8.110 GiNaC::partition\_with\_zero\_parts\_generator Class Reference

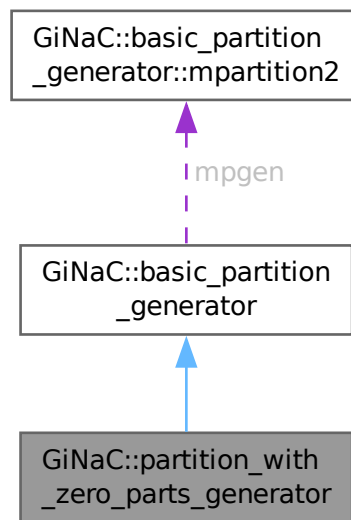
Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::partition\_with\_zero\_parts\_generator:



Collaboration diagram for `GiNaC::partition_with_zero_parts_generator`:



### Public Member Functions

- [partition\\_with\\_zero\\_parts\\_generator](#) (unsigned `n_`, unsigned `m_`)
- `const std::vector< unsigned > & get () const`
- `bool next ()`

### Private Attributes

- unsigned `m`
- `std::vector< unsigned > partition`
- `bool current_updated`

### Additional Inherited Members

### Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)

- [basic\\_partition\\_generator](#) (unsigned `n_`, unsigned `m_`)

### Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)

- `mpartition2 mpgen`

## 8.110.1 Detailed Description

Generate all bounded combinatorial partitions of an integer `n` with exactly `m` parts (including zero parts) in non-decreasing order.

## 8.110.2 Constructor & Destructor Documentation

### 8.110.2.1 partition\_with\_zero\_parts\_generator()

```
GiNaC::partition_with_zero_parts_generator::partition_with_zero_parts_generator (
    unsigned n_,
    unsigned m_) [inline]
```

References [GiNaC::basic\\_partition\\_generator::basic\\_partition\\_generator\(\)](#), [current\\_updated](#), [m](#), and [partition](#).

## 8.110.3 Member Function Documentation

### 8.110.3.1 get()

```
const std::vector< unsigned > & GiNaC::partition_with_zero_parts_generator::get () const
[inline]
```

References [current\\_updated](#), [m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), and [partition](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

### 8.110.3.2 next()

```
bool GiNaC::partition_with_zero_parts_generator::next () [inline]
```

References [current\\_updated](#), [m](#), and [GiNaC::basic\\_partition\\_generator::mpgen](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

## 8.110.4 Member Data Documentation

### 8.110.4.1 m

```
unsigned GiNaC::partition_with_zero_parts_generator::m [private]
```

Referenced by [get\(\)](#), [next\(\)](#), and [partition\\_with\\_zero\\_parts\\_generator\(\)](#).

### 8.110.4.2 partition

```
std::vector<unsigned> GiNaC::partition_with_zero_parts_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#), and [partition\\_with\\_zero\\_parts\\_generator\(\)](#).

### 8.110.4.3 current\_updated

```
bool GiNaC::partition_with_zero_parts_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), [next\(\)](#), and [partition\\_with\\_zero\\_parts\\_generator\(\)](#).

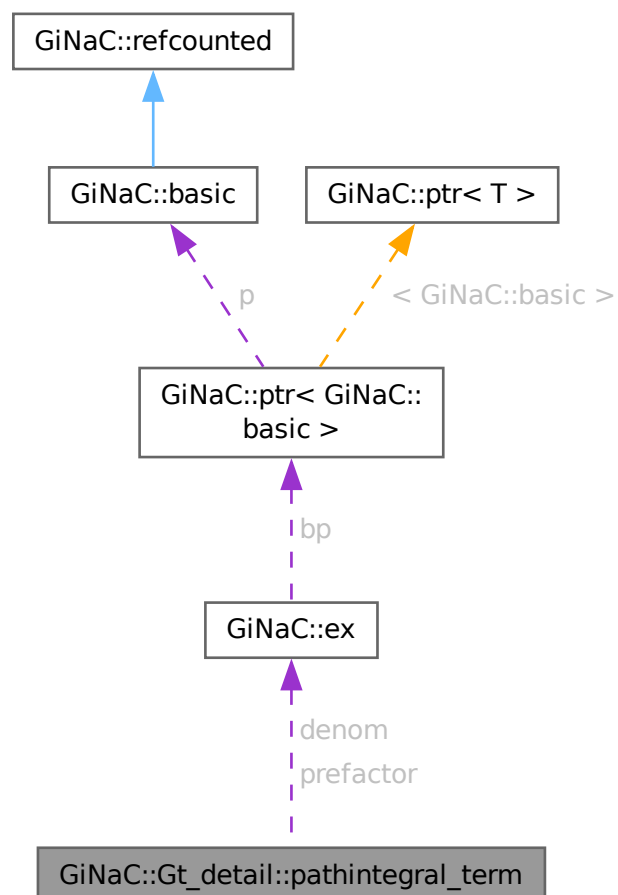
The documentation for this class was generated from the following file:

- [utils.h](#)

## 8.111 GiNaC::Gt\_detail::pathintegral\_term Class Reference

```
#include <Gt_helpers.h>
```

Collaboration diagram for GiNaC::Gt\_detail::pathintegral\_term:



## Public Member Functions

- [pathintegral\\_term](#) ([ex prefactor](#), [int power](#), [const std::vector< ex > &polylog={}](#), [ex denom=0](#))
- [pathintegral\\_term](#) ([const pathintegral\\_term &a](#), [const pathintegral\\_term &b](#), [int power\\_offset=0](#))
- [void integrate](#) ([const ex &start](#), [pathintegral\\_term\\_list &integrand](#), [pathintegral\\_term\\_list &result](#)) [const](#)
- [ex evaluate](#) ([const ex &upper\\_bound](#), [const std::vector< ex > &path](#)) [const](#)

## Static Private Member Functions

- [static ex G\\_path](#) ([const std::vector< ex > &args](#), [const std::vector< ex > &path](#))
- [static ex G\\_path](#) ([const std::vector< ex > &args](#), [const ex &start](#), [const ex &end](#))

## Private Attributes

- [ex prefactor](#)
- [int power](#)
- [std::vector< ex > polylog](#)
- [ex denom](#)

## Friends

- [class pathintegral\\_term\\_list](#)
- [std::ostream & operator<<](#) ([std::ostream &os](#), [const pathintegral\\_term &term](#))
- [bool operator<](#) ([const pathintegral\\_term &lh](#), [const pathintegral\\_term &rh](#))

## 8.111.1 Constructor & Destructor Documentation

### 8.111.1.1 pathintegral\_term() [1/2]

```
GiNaC::Gt_detail::pathintegral_term::pathintegral_term (
    ex prefactor,
    int power,
    const std::vector< ex > & polylog = {},
    ex denom = 0) \[inline\]
```

References [polylog](#), [power](#), and [prefactor](#).

Referenced by [operator<](#), [operator<<](#), [pathintegral\\_term\(\)](#), and [pathintegral\\_term\\_list](#).

### 8.111.1.2 pathintegral\_term() [2/2]

```
GiNaC::Gt_detail::pathintegral_term::pathintegral_term (
    const pathintegral\_term & a,
    const pathintegral\_term & b,
    int power\_offset = 0)
```

References [denom](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [pathintegral\\_term\(\)](#), [polylog](#), [power](#), and [prefactor](#).

## 8.111.2 Member Function Documentation

### 8.111.2.1 `integrate()`

```
void GiNaC::Gt_detail::pathintegral_term::integrate (
    const ex & start,
    pathintegral_term_list & integrand,
    pathintegral_term_list & result) const
```

References [GiNaC::Gt\\_detail::pathintegral\\_term\\_list::add\(\)](#), [GiNaC::class\\_info< OPT >::first](#), [denom](#), [pathintegral\\_term\\_list](#), [polylog](#), [GiNaC::pow\(\)](#), [power](#), and [prefactor](#).

### 8.111.2.2 `evaluate()`

```
ex GiNaC::Gt_detail::pathintegral_term::evaluate (
    const ex & upper_bound,
    const std::vector< ex > & path) const
```

References [denom](#), [G\\_path\(\)](#), [polylog](#), [GiNaC::pow\(\)](#), [power](#), and [prefactor](#).

### 8.111.2.3 `G_path()` [1/2]

```
ex GiNaC::Gt_detail::pathintegral_term::G_path (
    const std::vector< ex > & args,
    const std::vector< ex > & path) [static], [private]
```

References [GiNaC::Gt\\_detail::deconcatenate\\_path< ex >\(\)](#), and [G\\_path\(\)](#).

Referenced by [evaluate\(\)](#), and [G\\_path\(\)](#).

### 8.111.2.4 `G_path()` [2/2]

```
ex GiNaC::Gt_detail::pathintegral_term::G_path (
    const std::vector< ex > & args,
    const ex & start,
    const ex & end) [static], [private]
```

References [GiNaC::container< C >::append\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::G\(\)](#), [GiNaC::basic::hold\(\)](#), and [GiNaC::pow\(\)](#).

## 8.111.3 Friends And Related Symbol Documentation

### 8.111.3.1 `pathintegral_term_list`

```
friend class pathintegral_term_list [friend]
```

References [pathintegral\\_term\(\)](#), and [pathintegral\\_term\\_list](#).

Referenced by [integrate\(\)](#), and [pathintegral\\_term\\_list](#).



### 8.111.3.2 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const pathintegral_term & term) [friend]
```

References [denom](#), [GiNaC::ex::is\\_zero\(\)](#), [pathintegral\\_term\(\)](#), [polylog](#), [power](#), and [prefactor](#).

### 8.111.3.3 operator<

```
bool operator< (
    const pathintegral_term & lh,
    const pathintegral_term & rh) [friend]
```

References [GiNaC::ex::compare\(\)](#), [denom](#), [GiNaC::ex::is\\_zero\(\)](#), [pathintegral\\_term\(\)](#), [polylog](#), and [power](#).

## 8.111.4 Member Data Documentation

### 8.111.4.1 prefactor

```
ex GiNaC::Gt_detail::pathintegral_term::prefactor [mutable], [private]
```

Referenced by [evaluate\(\)](#), [integrate\(\)](#), [operator<<](#), [pathintegral\\_term\(\)](#), and [pathintegral\\_term\(\)](#).

### 8.111.4.2 power

```
int GiNaC::Gt_detail::pathintegral_term::power [private]
```

Referenced by [evaluate\(\)](#), [integrate\(\)](#), [operator<](#), [operator<<](#), [pathintegral\\_term\(\)](#), and [pathintegral\\_term\(\)](#).

### 8.111.4.3 polylog

```
std::vector<ex> GiNaC::Gt_detail::pathintegral_term::polylog [private]
```

Referenced by [evaluate\(\)](#), [integrate\(\)](#), [operator<](#), [operator<<](#), [pathintegral\\_term\(\)](#), and [pathintegral\\_term\(\)](#).

### 8.111.4.4 denom

```
ex GiNaC::Gt_detail::pathintegral_term::denom [private]
```

Referenced by [evaluate\(\)](#), [integrate\(\)](#), [operator<](#), [operator<<](#), and [pathintegral\\_term\(\)](#).

The documentation for this class was generated from the following files:

- [Gt\\_helpers.h](#)
- [Gt\\_helpers.cpp](#)

## 8.112 GiNaC::Gt\_detail::pathintegral\_term\_list Class Reference

```
#include <Gt_helpers.h>
```

### Public Member Functions

- `std::set< pathintegral\_term >::const_iterator begin ()` const
- `std::set< pathintegral\_term >::const_iterator end ()` const
- `void clear ()`
- `bool empty ()` const
- `size_t size ()` const
- `void add (pathintegral\_term &&term)`
- `template<typename... Args, typename = typename std::enable_if<(sizeof...(Args)>1)>::type>`  
`void add (Args &&...args)`
- `void add (const pathintegral\_term\_list &other)`
- `pathintegral\_term pop ()`

### Private Attributes

- `std::set< pathintegral\_term > terms`

## 8.112.1 Member Function Documentation

### 8.112.1.1 `begin()`

```
std::set< pathintegral\_term >::const_iterator GiNaC::Gt_detail::pathintegral_term_list::begin  
( ) const [inline]
```

References [terms](#).

Referenced by [pop\(\)](#).

### 8.112.1.2 `end()`

```
std::set< pathintegral\_term >::const_iterator GiNaC::Gt_detail::pathintegral_term_list::end ( )  
const [inline]
```

References [terms](#).

Referenced by [add\(\)](#).

### 8.112.1.3 `clear()`

```
void GiNaC::Gt_detail::pathintegral_term_list::clear ( ) [inline]
```

References [terms](#).

#### 8.112.1.4 empty()

```
bool GiNaC::Gt_detail::pathintegral_term_list::empty () const [inline]
```

References [terms](#).

#### 8.112.1.5 size()

```
size_t GiNaC::Gt_detail::pathintegral_term_list::size () const [inline]
```

References [terms](#).

#### 8.112.1.6 add() [1/3]

```
void GiNaC::Gt_detail::pathintegral_term_list::add (  
    pathintegral_term && term)
```

References [end\(\)](#), and [terms](#).

Referenced by [add\(\)](#), [add\(\)](#), and [GiNaC::Gt\\_detail::pathintegral\\_term::integrate\(\)](#).

#### 8.112.1.7 add() [2/3]

```
template<typename... Args, typename = typename std::enable_if<(sizeof...(Args)>1)>::type>  
void GiNaC::Gt_detail::pathintegral_term_list::add (  
    Args &&... args) [inline]
```

References [add\(\)](#).

#### 8.112.1.8 add() [3/3]

```
void GiNaC::Gt_detail::pathintegral_term_list::add (  
    const pathintegral_term_list & other)
```

References [add\(\)](#).

#### 8.112.1.9 pop()

```
pathintegral_term GiNaC::Gt_detail::pathintegral_term_list::pop ()
```

References [begin\(\)](#), and [terms](#).

## 8.112.2 Member Data Documentation

### 8.112.2.1 terms

```
std::set<pathintegral_term> GiNaC::Gt_detail::pathintegral_term_list::terms [private]
```

Referenced by [add\(\)](#), [begin\(\)](#), [clear\(\)](#), [empty\(\)](#), [end\(\)](#), [pop\(\)](#), and [size\(\)](#).

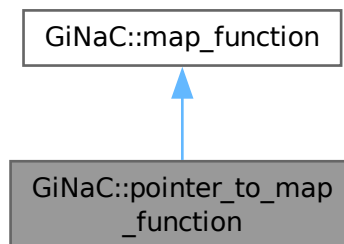
The documentation for this class was generated from the following files:

- [Gt\\_helpers.h](#)
- [Gt\\_helpers.cpp](#)

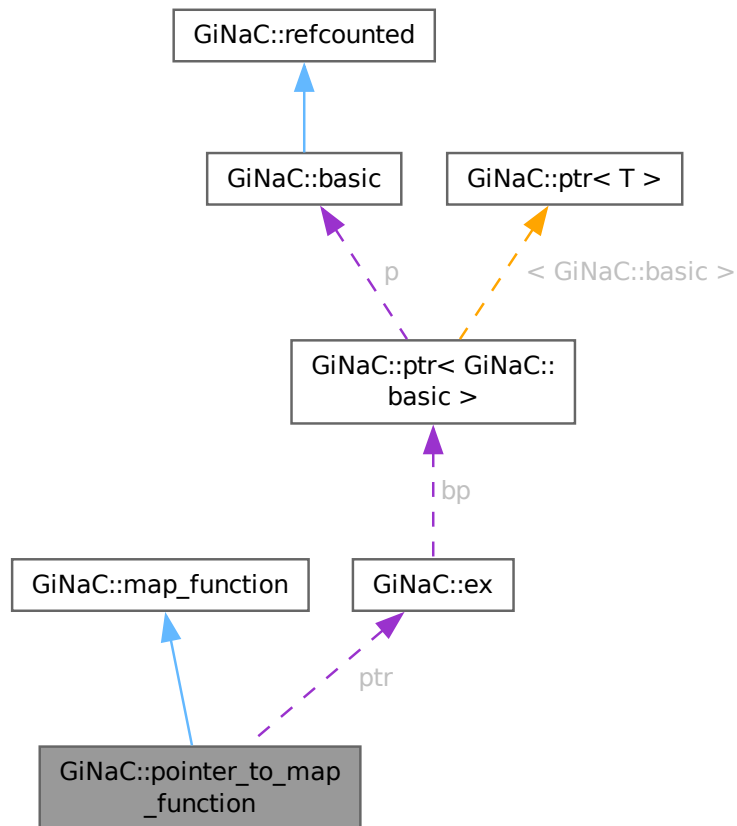
## 8.113 GiNaC::pointer\_to\_map\_function Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function:



Collaboration diagram for GiNaC::pointer\_to\_map\_function:



### Public Member Functions

- [pointer\\_to\\_map\\_function](#) ([ex](#) x(const [ex](#) &))
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &)

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 8.113.1 Constructor & Destructor Documentation

#### 8.113.1.1 `pointer_to_map_function()`

```
GiNaC::pointer_to_map_function::pointer_to_map_function (
    ex xconst ex &) [inline], [explicit]
```

References [ptr](#).

### 8.113.2 Member Function Documentation

#### 8.113.2.1 `operator>()()`

```
ex GiNaC::pointer_to_map_function::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [ptr](#).

### 8.113.3 Member Data Documentation

#### 8.113.3.1 `ptr`

```
ex(* GiNaC::pointer_to_map_function::ptr) (const ex &) [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_map\\_function\(\)](#).

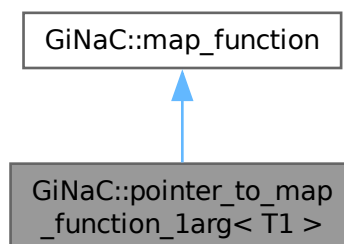
The documentation for this class was generated from the following file:

- [ex.h](#)

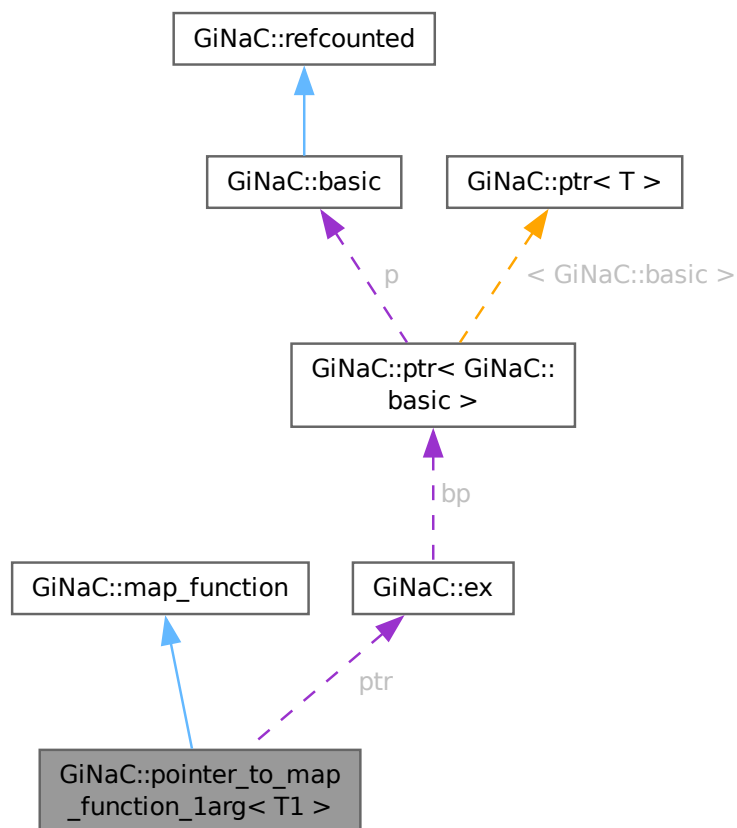
## 8.114 `GiNaC::pointer_to_map_function_1arg< T1 >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_map_function_1arg< T1 >`:



Collaboration diagram for GiNaC::pointer\_to\_map\_function\_1arg< T1 >:



### Public Member Functions

- `pointer_to_map_function_1arg` (`ex` x(const `ex` &, T1), T1 a1)
- `ex operator()` (const `ex` &e) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function` ()

### Protected Attributes

- `ex(* ptr)`(const `ex` &, T1)
- T1 `arg1`

### Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

## 8.114.1 Constructor & Destructor Documentation

### 8.114.1.1 `pointer_to_map_function_1arg()`

```
template<class T1>
GiNaC::pointer_to_map_function_1arg< T1 >::pointer_to_map_function_1arg (
    ex xconst ex &, T1,
    T1 a1) [inline], [explicit]
```

References [arg1](#), and [ptr](#).

## 8.114.2 Member Function Documentation

### 8.114.2.1 `operator>()`

```
template<class T1>
ex GiNaC::pointer_to_map_function_1arg< T1 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), and [ptr](#).

## 8.114.3 Member Data Documentation

### 8.114.3.1 `ptr`

```
template<class T1>
ex(* GiNaC::pointer_to_map_function_1arg< T1 >::ptr) (const ex &, T1) [protected]
```

Referenced by [operator>\(\)](#), and [pointer\\_to\\_map\\_function\\_1arg\(\)](#).

### 8.114.3.2 `arg1`

```
template<class T1>
T1 GiNaC::pointer_to_map_function_1arg< T1 >::arg1 [protected]
```

Referenced by [operator>\(\)](#), and [pointer\\_to\\_map\\_function\\_1arg\(\)](#).

The documentation for this class was generated from the following file:

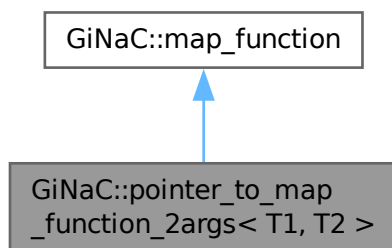
- [ex.h](#)



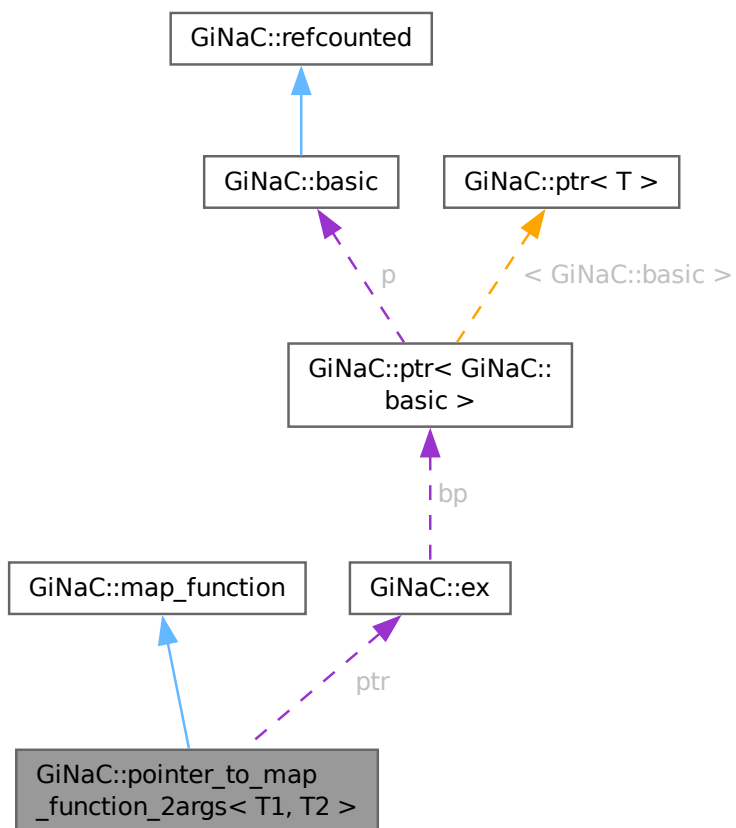
## 8.115 GiNaC::pointer\_to\_map\_function\_2args< T1, T2 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



Collaboration diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



## Public Member Functions

- [pointer\\_to\\_map\\_function\\_2args](#) ([ex](#) x(const [ex](#) &, T1, T2), T1 a1, T2 a2)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &, T1, T2)
- T1 [arg1](#)
- T2 [arg2](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.115.1 Constructor & Destructor Documentation

### 8.115.1.1 [pointer\\_to\\_map\\_function\\_2args](#)()

```
template<class T1, class T2>
GiNaC::pointer_to_map_function_2args< T1, T2 >::pointer_to_map_function_2args (
    ex xconst ex &, T1, T2,
    T1 a1,
    T2 a2) [inline], [explicit]
```

References [arg1](#), [arg2](#), and [ptr](#).

## 8.115.2 Member Function Documentation

### 8.115.2.1 [operator](#)()()

```
template<class T1, class T2>
ex GiNaC::pointer_to_map_function_2args< T1, T2 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), [arg2](#), and [ptr](#).

### 8.115.3 Member Data Documentation

#### 8.115.3.1 ptr

```
template<class T1, class T2>
ex(* GiNaC::pointer_to_map_function_2args< T1, T2 >::ptr) (const ex &, T1, T2) [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_2args\(\)](#).

#### 8.115.3.2 arg1

```
template<class T1, class T2>
T1 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg1 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_2args\(\)](#).

#### 8.115.3.3 arg2

```
template<class T1, class T2>
T2 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg2 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_2args\(\)](#).

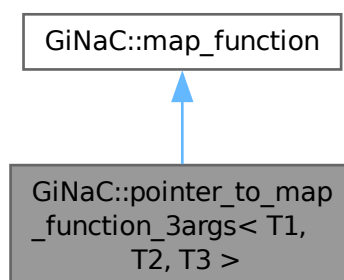
The documentation for this class was generated from the following file:

- [ex.h](#)

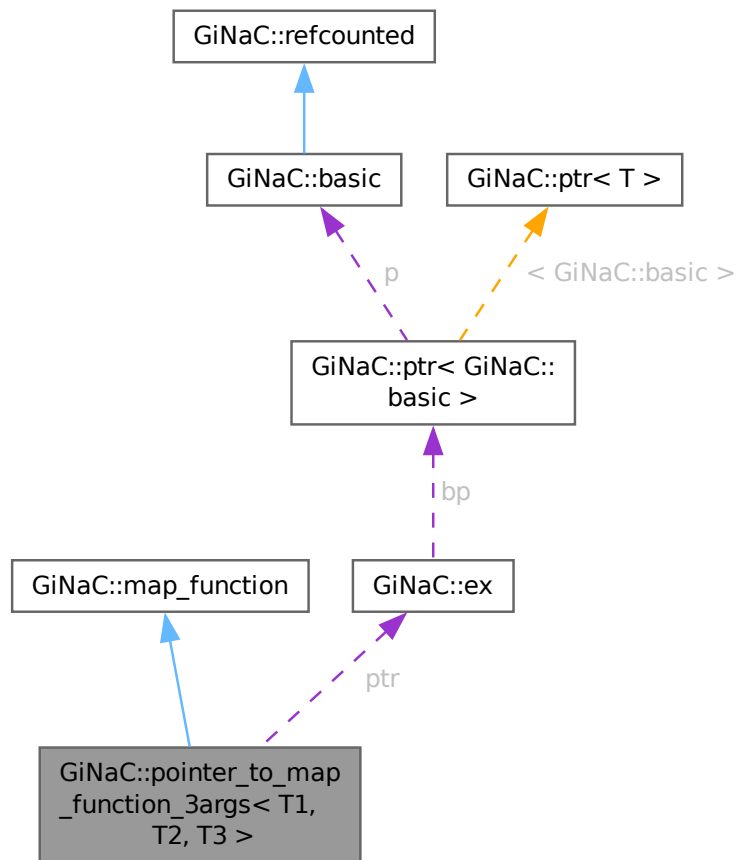
## 8.116 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >:



Collaboration diagram for `GiNaC::pointer_to_map_function_3args< T1, T2, T3 >`:



### Public Member Functions

- `pointer_to_map_function_3args` (`ex` x(const `ex` &, T1, T2, T3), T1 a1, T2 a2, T3 a3)
- `ex operator()` (const `ex` &e) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`

### Protected Attributes

- `ex(* ptr)`(const `ex` &, T1, T2, T3)
- T1 `arg1`
- T2 `arg2`
- T3 `arg3`

## Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.116.1 Constructor & Destructor Documentation

### 8.116.1.1 [pointer\\_to\\_map\\_function\\_3args\(\)](#)

```
template<class T1, class T2, class T3>
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::pointer_to_map_function_3args (
    ex xconst ex &, T1, T2, T3,
    T1 a1,
    T2 a2,
    T3 a3) [inline], [explicit]
```

References [arg1](#), [arg2](#), [arg3](#), and [ptr](#).

## 8.116.2 Member Function Documentation

### 8.116.2.1 [operator>\(\)\(\)](#)

```
template<class T1, class T2, class T3>
ex GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), [arg2](#), [arg3](#), and [ptr](#).

## 8.116.3 Member Data Documentation

### 8.116.3.1 [ptr](#)

```
template<class T1, class T2, class T3>
ex(* GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::ptr) (const ex &, T1, T2, T3) [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_3args\(\)](#).

### 8.116.3.2 [arg1](#)

```
template<class T1, class T2, class T3>
T1 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::arg1 [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_3args\(\)](#).

### 8.116.3.3 `arg2`

```
template<class T1, class T2, class T3>  
T2 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg2 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_3args\(\)](#).

### 8.116.3.4 `arg3`

```
template<class T1, class T2, class T3>  
T3 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg3 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_map\\_function\\_3args\(\)](#).

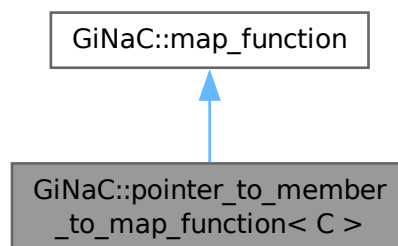
The documentation for this class was generated from the following file:

- [ex.h](#)

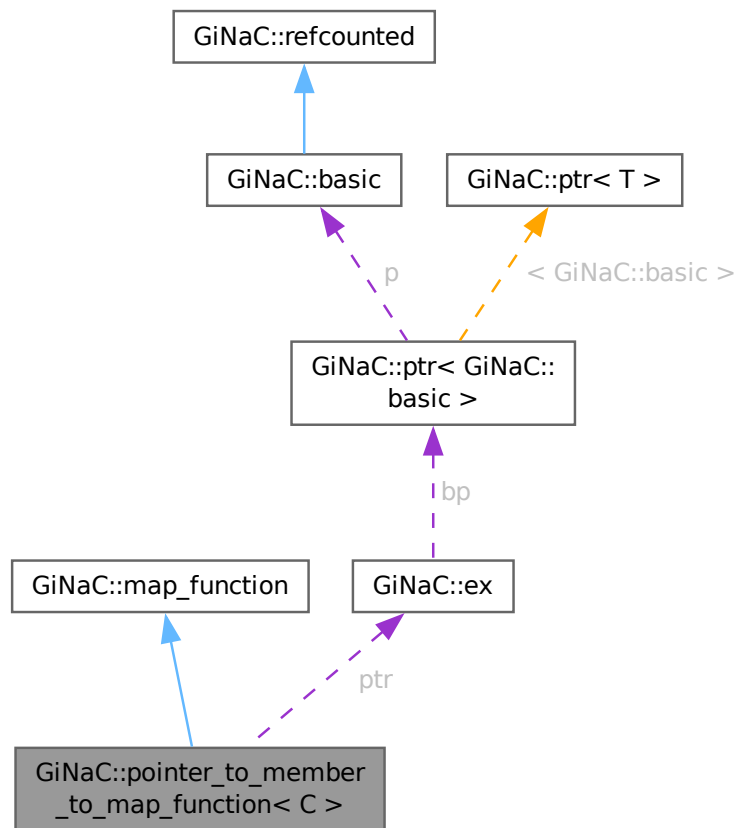
## 8.117 `GiNaC::pointer_to_member_to_map_function< C >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function< C >`:



Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function< C >:



### Public Member Functions

- `pointer_to_member_to_map_function` (`ex`(`C::*member`)(`const ex &`), `C &obj`)
- `ex operator()` (`const ex &e`) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function` ()

### Protected Attributes

- `ex`(`C::* ptr`)(`const ex &`)
- `C &c`

### Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef `const ex & argument_type`
- typedef `ex result_type`

## 8.117.1 Constructor & Destructor Documentation

### 8.117.1.1 `pointer_to_member_to_map_function()`

```
template<class C>
GiNaC::pointer_to_member_to_map_function< C >::pointer_to_member_to_map_function (
    ex(C::* member ) (const ex &),
    C & obj) [inline], [explicit]
```

References [c](#), and [ptr](#).

## 8.117.2 Member Function Documentation

### 8.117.2.1 `operator>()`

```
template<class C>
ex GiNaC::pointer_to_member_to_map_function< C >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [c](#), and [ptr](#).

## 8.117.3 Member Data Documentation

### 8.117.3.1 `ptr`

```
template<class C>
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function< C >::ptr) (const ex &) [protected]
```

Referenced by [operator>\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\(\)](#).

### 8.117.3.2 `c`

```
template<class C>
C& GiNaC::pointer\_to\_member\_to\_map\_function< C >::c [protected]
```

Referenced by [operator>\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\(\)](#).

The documentation for this class was generated from the following file:

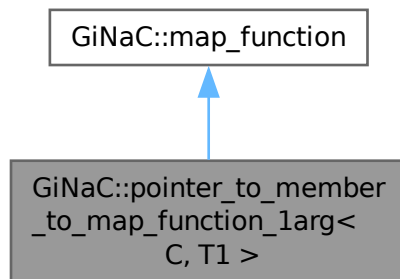
- [ex.h](#)



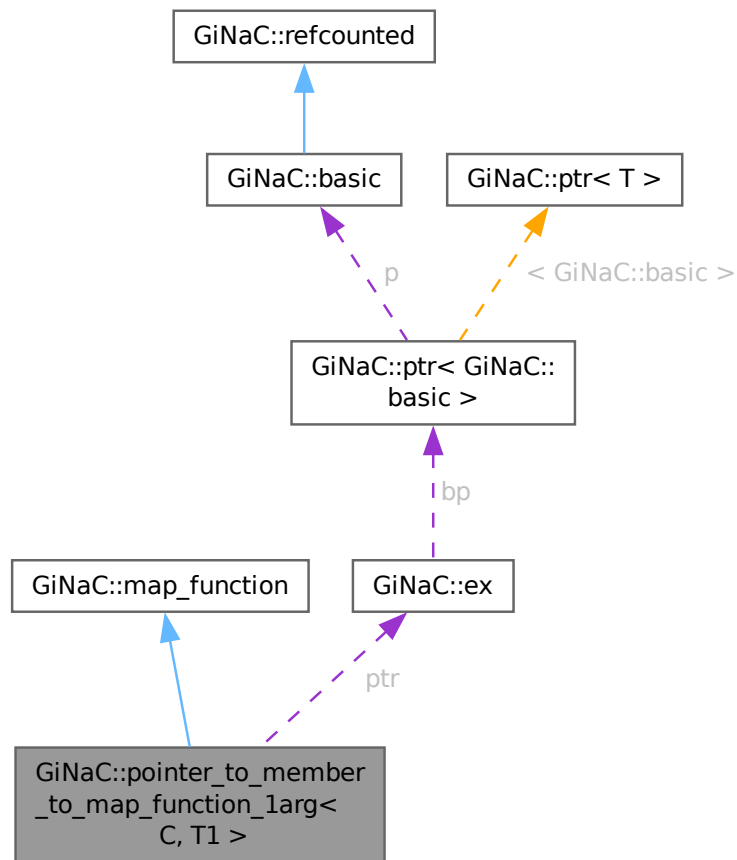
## 8.118 GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >:



Collaboration diagram for `GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >`:



### Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#) ([ex](#)(C::[\\*member](#))(const [ex](#) &, T1), C &obj, T1 a1)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(C::[\\* ptr](#))(const [ex](#) &, T1)
- C & [c](#)
- T1 [arg1](#)

## Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.118.1 Constructor & Destructor Documentation

### 8.118.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#)

```
template<class C, class T1>
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::pointer_to_member_to_map_function_1arg
(
    ex(C::* member ) (const ex &, T1),
    C & obj,
    T1 a1) [inline], [explicit]
```

References [arg1](#), [c](#), and [ptr](#).

## 8.118.2 Member Function Documentation

### 8.118.2.1 [operator>\(\)\(\)](#)

```
template<class C, class T1>
ex GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), [c](#), and [ptr](#).

## 8.118.3 Member Data Documentation

### 8.118.3.1 [ptr](#)

```
template<class C, class T1>
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >::ptr) (const ex &, T1) [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#).

### 8.118.3.2 [c](#)

```
template<class C, class T1>
C& GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >::c [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#).

### 8.118.3.3 `arg1`

```
template<class C, class T1>  
T1 GiNaC::pointer_to_member_to_map_function_larg< C, T1 >::arg1 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#).

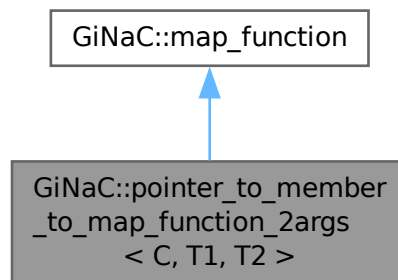
The documentation for this class was generated from the following file:

- [ex.h](#)

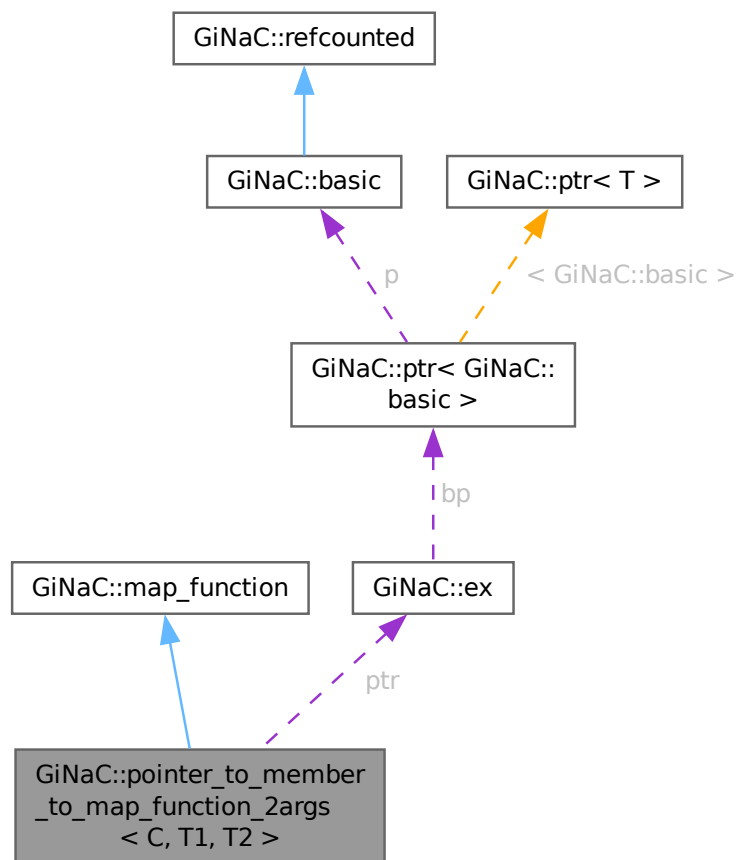
## 8.119 `GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >`:



Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >:



### Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#) ([ex](#)(C::\*member)(const [ex](#) &, T1, T2), C &obj, T1 a1, T2 a2)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(C::\* [ptr](#))(const [ex](#) &, T1, T2)
- C & [c](#)
- T1 [arg1](#)
- T2 [arg2](#)

## Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.119.1 Constructor & Destructor Documentation

### 8.119.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#)

```
template<class C, class T1, class T2>
GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::pointer_to_member_to_map_function_2args (
    ex(C::* member) (const ex &, T1, T2),
    C & obj,
    T1 a1,
    T2 a2) [inline], [explicit]
```

References [arg1](#), [arg2](#), [c](#), and [ptr](#).

## 8.119.2 Member Function Documentation

### 8.119.2.1 [operator>\(\)\(\)](#)

```
template<class C, class T1, class T2>
ex GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), [arg2](#), [c](#), and [ptr](#).

## 8.119.3 Member Data Documentation

### 8.119.3.1 [ptr](#)

```
template<class C, class T1, class T2>
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::ptr) (const ex &, T1, T2)
[protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#).

### 8.119.3.2 [c](#)

```
template<class C, class T1, class T2>
C& GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::c [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#).

## 8.119.3.3 arg1

```
template<class C, class T1, class T2>
T1 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg1 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#).

## 8.119.3.4 arg2

```
template<class C, class T1, class T2>
T2 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg2 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#).

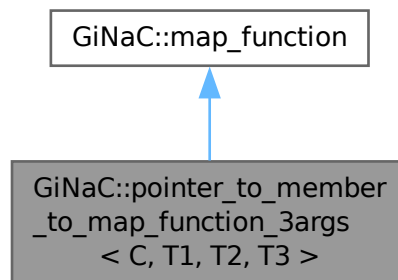
The documentation for this class was generated from the following file:

- [ex.h](#)

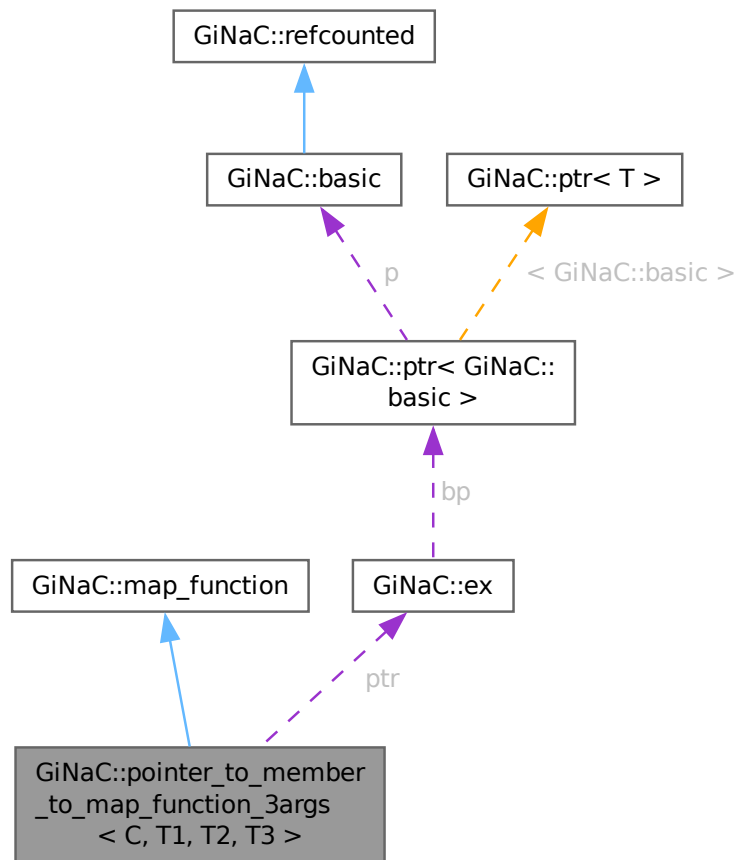
## 8.120 GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >:



Collaboration diagram for `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >`:



### Public Member Functions

- `pointer_to_member_to_map_function_3args` (`ex(C::*member)(const ex &, T1, T2, T3), C &obj, T1 a1, T2 a2, T3 a3`)
- `ex operator()` (`const ex &e`) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`

### Protected Attributes

- `ex(C::* ptr)` (`const ex &, T1, T2, T3`)
- `C & c`
- `T1 arg1`
- `T2 arg2`
- `T3 arg3`



## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 8.120.1 Constructor & Destructor Documentation

### 8.120.1.1 pointer\_to\_member\_to\_map\_function\_3args()

```
template<class C, class T1, class T2, class T3>
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::pointer_to_member_to_map_↵
function_3args (
    ex(C::* member )(const ex &, T1, T2, T3),
    C & obj,
    T1 a1,
    T2 a2,
    T3 a3) [inline], [explicit]
```

References [arg1](#), [arg2](#), [arg3](#), [c](#), and [ptr](#).

## 8.120.2 Member Function Documentation

### 8.120.2.1 operator>()()

```
template<class C, class T1, class T2, class T3>
ex GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator() (
    const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [arg1](#), [arg2](#), [arg3](#), [c](#), and [ptr](#).

## 8.120.3 Member Data Documentation

### 8.120.3.1 ptr

```
template<class C, class T1, class T2, class T3>
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >::ptr) (const ex &, T1,
T2, T3) [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#).

### 8.120.3.2 c

```
template<class C, class T1, class T2, class T3>
C& GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >::c [protected]
```

Referenced by [operator>\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#).

### 8.120.3.3 `arg1`

```
template<class C, class T1, class T2, class T3>
T1 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg1 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#).

### 8.120.3.4 `arg2`

```
template<class C, class T1, class T2, class T3>
T2 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg2 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#).

### 8.120.3.5 `arg3`

```
template<class C, class T1, class T2, class T3>
T3 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg3 [protected]
```

Referenced by [operator\(\)\(\)](#), and [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#).

The documentation for this class was generated from the following file:

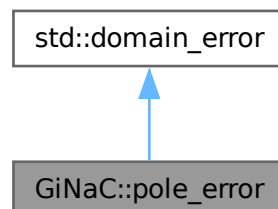
- [ex.h](#)

## 8.121 `GiNaC::pole_error` Class Reference

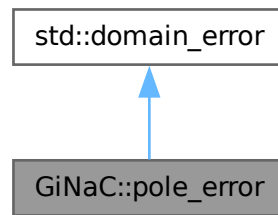
Exception class thrown when a singularity is encountered.

```
#include <numeric.h>
```

Inheritance diagram for `GiNaC::pole_error`:



Collaboration diagram for GiNaC::pole\_error:



### Public Member Functions

- `pole_error` (const std::string &what\_arg, int degree)  
*ctor for `pole_error` exception class.*
- int `degree` () const  
*Return the degree of the `pole_error` exception class.*

### Private Attributes

- int `deg`

## 8.121.1 Detailed Description

Exception class thrown when a singularity is encountered.

## 8.121.2 Constructor & Destructor Documentation

### 8.121.2.1 pole\_error()

```
GiNaC::pole_error::pole_error (  
    const std::string & what_arg,  
    int degree) [explicit]
```

ctor for `pole_error` exception class.

References `deg`, and `degree()`.

## 8.121.3 Member Function Documentation

### 8.121.3.1 degree()

```
int GiNaC::pole_error::degree () const
```

Return the degree of the `pole_error` exception class.

References `deg`.

Referenced by `pole_error()`.

## 8.121.4 Member Data Documentation

### 8.121.4.1 deg

```
int GiNaC::pole_error::deg [private]
```

Referenced by [degree\(\)](#), and [pole\\_error\(\)](#).

The documentation for this class was generated from the following files:

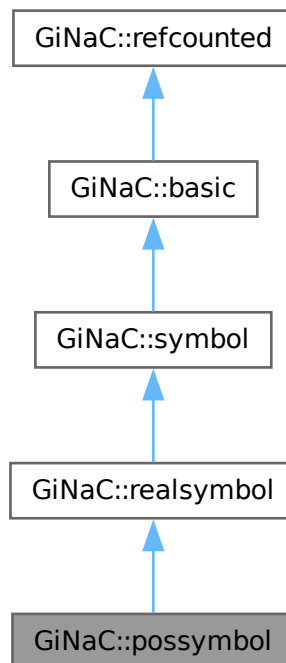
- [numeric.h](#)
- [utils.cpp](#)

## 8.122 GiNaC::possymbol Class Reference

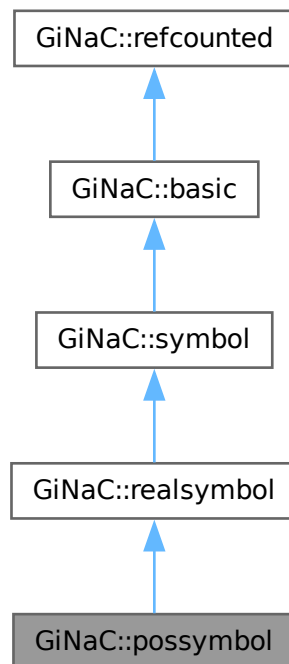
Specialization of symbol to real positive domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::possymbol:



Collaboration diagram for GiNaC::possymbol:



### Public Member Functions

- [possymbol](#) ()
- [possymbol](#) (const std::string &initname)
- [possymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [possymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

### Public Member Functions inherited from [GiNaC::realsymbol](#)

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

## Public Member Functions inherited from `GiNaC::symbol`

- `symbol` (const std::string &initname)
- `symbol` (const std::string &initname, const std::string &texname)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- `ex series` (const `relational` &s, int order, unsigned options=0) const override  
*Implementation of `ex::series()` for symbols.*
- `ex subs` (const `exmap` &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Implementation of `ex::normal()` for symbols.*
- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for symbols.*
- `ex to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for symbols.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- void `set_name` (const std::string &n)
- void `set_TeX_name` (const std::string &n)
- std::string `get_name` () const
- std::string `get_TeX_name` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const basic &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*

- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation ex::max\_coefficient().*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*

- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Additional Inherited Members

## Protected Member Functions inherited from `GiNaC::symbol`

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- `bool is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual bool match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*



## Protected Attributes inherited from [GiNaC::symbol](#)

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printrname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.122.1 Detailed Description

Specialization of symbol to real positive domain.

### 8.122.2 Constructor & Destructor Documentation

#### 8.122.2.1 `possymbol()` [1/3]

```
GiNaC::possymbol::possymbol ()
```

References [GiNaC::realsymbol::realsymbol\(\)](#).

Referenced by [duplicate\(\)](#).

#### 8.122.2.2 `possymbol()` [2/3]

```
GiNaC::possymbol::possymbol (  
    const std::string & initname) [explicit]
```

References [GiNaC::realsymbol::realsymbol\(\)](#).

#### 8.122.2.3 `possymbol()` [3/3]

```
GiNaC::possymbol::possymbol (  
    const std::string & initname,  
    const std::string & texname)
```

References [GiNaC::realsymbol::realsymbol\(\)](#).

### 8.122.3 Member Function Documentation

#### 8.122.3.1 `get_domain()`

```
unsigned GiNaC::possymbol::get_domain () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

References [GiNaC::domain::positive](#).

#### 8.122.3.2 `duplicate()`

```
possymbol * GiNaC::possymbol::duplicate () const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::dynallocated](#), [possymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

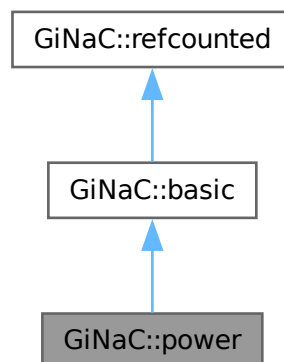
- [symbol.h](#)
- [symbol.cpp](#)

## 8.123 GiNaC::power Class Reference

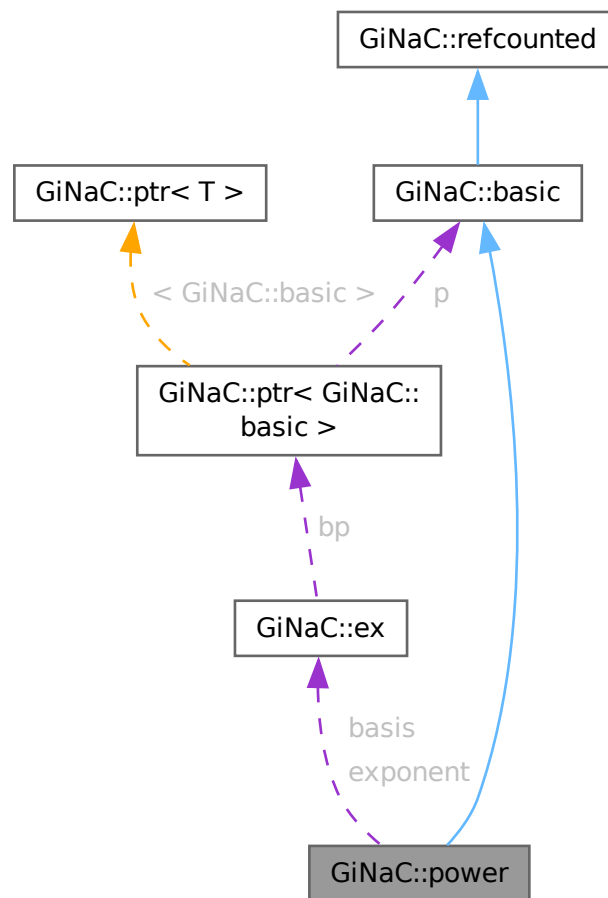
This class holds a two-component object, a basis and an exponent representing exponentiation.

```
#include <power.h>
```

Inheritance diagram for GiNaC::power:



Collaboration diagram for GiNaC::power:



## Public Member Functions

- `power` (const `ex` &lh, const `ex` &rh)
- `template<typename T>`  
`power` (const `ex` &lh, const T &rh)
- `unsigned precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- `bool info` (unsigned inf) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `bool is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*

- int **degree** (const **ex** &s) const override  
*Return degree of highest power in object s.*
- int **ldegree** (const **ex** &s) const override  
*Return degree of lowest power in object s.*
- **ex coeff** (const **ex** &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- **ex eval** () const override  
*Perform automatic term rewriting rules in this class.*
- **ex evalf** () const override  
*Evaluate object numerically.*
- **ex evalm** () const override  
*Evaluate sums, products and integer powers of matrices.*
- **ex series** (const **relational** &s, int order, unsigned options=0) const override  
*Implementation of **ex::series()** for powers.*
- **ex subs** (const **exmap** &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- bool **has** (const **ex** &other, unsigned options=0) const override  
*Test for occurrence of a pattern.*
- **ex normal** (**exmap** &repl, **exmap** &rev\_lookup, **lst** &modifier) const override  
*Implementation of **ex::normal()** for powers.*
- **ex to\_rational** (**exmap** &repl) const override  
*Implementation of **ex::to\_rational()** for powers.*
- **ex to\_polynomial** (**exmap** &repl) const override  
*Implementation of **ex::to\_polynomial()** for powers.*
- **ex conjugate** () const override
- **ex real\_part** () const override
- **ex imag\_part** () const override
- void **archive** (**archive\_node** &n) const override  
*Save (a.k.a.*
- void **read\_archive** (const **archive\_node** &n, **lst** &symbols) override  
*Read (a.k.a.*

## Public Member Functions inherited from **GiNaC::basic**

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const **basic** &other)
- const **basic** & **operator=** (const **basic** &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic** \* **duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const **basic** &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const **print\_context** &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const

- Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)
- Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a power.*
- `ex eval_ncmul` (const `exvector` &v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- void `print_power` (const `print_context` &c, const char \*powersymbol, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print_dflt` (const `print_dflt` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- void `do_print_csrc_cl_N` (const `print_csrc_cl_N` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Static Protected Member Functions

- static `ex expand_add` (const `add` &a, long n, unsigned options)  
*expand  $a^n$  where  $a$  is an add and  $n$  is a positive integer.*
- static `ex expand_add_2` (const `add` &a, unsigned options)  
*Special case of `power::expand_add`.*
- static `ex expand_mul` (const `mul` &m, const `numeric` &n, unsigned options, bool from\_expand=false)  
*Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.*

## Protected Attributes

- `ex basis`
- `ex exponent`

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Friends

- class [mul](#)

## 8.123.1 Detailed Description

This class holds a two-component object, a basis and an exponent representing exponentiation.

## 8.123.2 Constructor & Destructor Documentation

### 8.123.2.1 [power\(\)](#) [1/2]

```
GiNaC::power::power (
    const ex & lh,
    const ex & rh) [inline]
```

References [basis](#), [GiNaC::basic::ex](#), [exponent](#), and [power\(\)](#).

Referenced by [do\\_print\\_latex\(\)](#), [eval\(\)](#), and [power\(\)](#).

### 8.123.2.2 [power\(\)](#) [2/2]

```
template<typename T>
GiNaC::power::power (
    const ex & lh,
    const T & rh) [inline]
```

References [basis](#), [GiNaC::basic::ex](#), and [exponent](#).

## 8.123.3 Member Function Documentation

### 8.123.3.1 [precedence\(\)](#)

```
unsigned GiNaC::power::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthesizing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_power\(\)](#).

### 8.123.3.2 info()

```
bool GiNaC::power::info (
    unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::basic::flags](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), and [GiNaC::basic::setflag\(\)](#).

### 8.123.3.3 nops()

```
size_t GiNaC::power::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.123.3.4 op()

```
ex GiNaC::power::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::basic::ex](#), [exponent](#), and [GINAC\\_ASSERT](#).

### 8.123.3.5 map()

```
ex GiNaC::power::map (
    map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), and [exponent](#).



### 8.123.3.6 is\_polynomial()

```
bool GiNaC::power::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::basic::ex](#), [exponent](#), and [GiNaC::info\\_flags::nonnegint](#).

### 8.123.3.7 degree()

```
int GiNaC::power::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::is\\_integer\(\)](#).

### 8.123.3.8 ldegree()

```
int GiNaC::power::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::is\\_integer\(\)](#).

### 8.123.3.9 coeff()

```
ex GiNaC::power::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [basis](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::is\\_integer\(\)](#).

Referenced by [expand\(\)](#), and [expand\\_add\(\)](#).

**8.123.3.10 eval()**

```
ex GiNaC::power::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x, x_1, x_2, \dots$  stand for a symbolic variables of type `ex` and  $c, c_1, c_2, \dots$  stand for such expressions that contain a plain number.

- $x^0 \rightarrow 1$  (also handles  $0^0$ )
- $x^1 \rightarrow x$
- $0^c \rightarrow 0$  or exception (depending on the real part of  $c$ )
- $1^x \rightarrow 1$
- $c_1^{c_2} \rightarrow c_1^{(c_2-n)} c_1^n$  (so that  $0 < (c_2-n) < 1$ , try to evaluate roots, possibly in numerator and denominator of  $c_1$ )
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_1 c_2)}$  if  $x$  is positive and  $c_1$  is real.
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_1 c_2)}$  ( $c_2$  integer or  $-1 < c_1 \leq 1$  or ( $c_1 = -1$  and  $c_2 > 0$ ), case  $c_1 = 1$  should not happen, see below!)
- $x^{(x, y, z), c} \rightarrow x^c y^c z^c$  (if  $c$  integer)
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_2)} c_1^{c_2}$  ( $c_1 > 0$ )
- $x^{(x, c_1), c_2} \rightarrow x^{(-x, c_2)} c_1^{c_2}$  ( $c_1 < 0$ )

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex0](#), [GiNaC::ex1](#), [GiNaC::ex\\_1](#), [GiNaC::num0\\_p](#), [GiNaC::num1\\_p](#), [GiNaC::num\\_1\\_p](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::numeric::compare\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expand\\_mul\(\)](#), [exponent](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::numeric::is\\_crational\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [likely](#), [GiNaC::numeric::mul\(\)](#), [mul](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [power\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::numeric::real\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

**8.123.3.11 evalf()**

```
ex GiNaC::power::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [exponent](#), and [GiNaC::is\\_exactly\\_a\(\)](#).

**8.123.3.12 evalm()**

```
ex GiNaC::power::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::pow\(\)](#).

**8.123.3.13 series()**

```
ex GiNaC::power::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for powers.

This performs Laurent expansion of reciprocals of series at singularities.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::ex::ldegree\(\)](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::to\\_int\(\)](#).

Referenced by [GiNaC::psi1\\_series\(\)](#).

**8.123.3.14 subs()**

```
ex GiNaC::power::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::pow\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

**8.123.3.15 has()**

```
bool GiNaC::power::has (
    const ex & pattern,
    unsigned options = 0) const \[override\], \[virtual\]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [basis](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::basic::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::ex::op\(\)](#), and [GiNaC::info\\_flags::posint](#).

**8.123.3.16 normal()**

```
ex GiNaC::power::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const \[override\], \[virtual\]
```

Implementation of [ex::normal\(\)](#) for powers.

It normalizes the basis, distributes integer exponents to numerator and denominator, and replaces non-integer powers by temporary symbols.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

**8.123.3.17 to\_rational()**

```
ex GiNaC::power::to_rational (
    exmap & repl) const \[override\], \[virtual\]
```

Implementation of [ex::to\\_rational\(\)](#) for powers.

It replaces non-integer powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::basic::ex](#), [exponent](#), [GiNaC::info\\_flags::integer](#), [GiNaC::pow\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**8.123.3.18 to\_polynomial()**

```
ex GiNaC::power::to_polynomial (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for powers.

It replaces non-posint powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::basic::ex](#), [exponent](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::posint](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::to\\_polynomial\(\)](#).

**8.123.3.19 conjugate()**

```
ex GiNaC::power::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [exponent](#), [GiNaC::info\\_flags::integer](#), and [GiNaC::info\\_flags::positive](#).

**8.123.3.20 real\_part()**

```
ex GiNaC::power::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), and [GiNaC::pow\(\)](#).

**8.123.3.21 imag\_part()**

```
ex GiNaC::power::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), and [GiNaC::sin\(\)](#).

**8.123.3.22 archive()**

```
void GiNaC::power::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [basis](#), and [exponent](#).

### 8.123.3.23 read\_archive()

```
void GiNaC::power::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [GiNaC::archive\\_node::find\\_ex\(\)](#).

### 8.123.3.24 derivative()

```
ex GiNaC::power::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [exponent](#), [GiNaC::is\\_a\(\)](#), [GiNaC::log\(\)](#), and [GiNaC::pow\(\)](#).

### 8.123.3.25 eval\_ncmul()

```
ex GiNaC::power::eval_ncmul (
    const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.123.3.26 return\_type()

```
unsigned GiNaC::power::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#).

**8.123.3.27 return\_type\_tinfo()**

```
return_type_t GiNaC::power::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#).

**8.123.3.28 expand()**

```
ex GiNaC::power::expand (
    unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_integer\(\)](#), [mul](#), [GiNaC::info\\_flags::negative](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#), [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

**8.123.3.29 print\_power()**

```
void GiNaC::power::print_power (
    const print_context & c,
    const char * powersymbol,
    const char * openbrace,
    const char * closebrace,
    unsigned level) const [protected]
```

References [basis](#), [exponent](#), [precedence\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

**8.123.3.30 do\_print\_dflt()**

```
void GiNaC::power::do_print_dflt (
    const print_dflt & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [exponent](#), [print\\_power\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.123.3.31 do\_print\_latex()**

```
void GiNaC::power::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [GiNaC::ex\\_to\(\)](#), [exponent](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_negative\(\)](#), [power\(\)](#), [print\\_power\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.123.3.32 do\_print\_csrc()**

```
void GiNaC::power::do_print_csrc (
    const print\_csrc & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.123.3.33 do\_print\_python()**

```
void GiNaC::power::do_print_python (
    const print\_python & c,
    unsigned level) const [protected]
```

References [print\\_power\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.123.3.34 do\_print\_python\_repr()**

```
void GiNaC::power::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [basis](#), [exponent](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.123.3.35 do\_print\_csrc\_cl\_N()**

```
void GiNaC::power::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [exponent](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).



**8.123.3.36 expand\_add()**

```
ex GiNaC::power::expand_add (
    const add & a,
    long n,
    unsigned options) [static], [protected]
```

expand  $a^n$  where  $a$  is an add and  $n$  is a positive integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [basis](#), [GiNaC::binomial\(\)](#), [coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [expand\\_add\\_2\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::factor\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [mul](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::composition\\_generator::next\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#).

**8.123.3.37 expand\_add\_2()**

```
ex GiNaC::power::expand_add_2 (
    const add & a,
    unsigned options) [static], [protected]
```

Special case of [power::expand\\_add](#).

Expands  $a^2$  where  $a$  is an add.

See also

[power::expand\\_add](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_num2\\_p](#), [basis](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [expand\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [mul](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

Referenced by [expand\\_add\(\)](#).

**8.123.3.38 expand\_mul()**

```
ex GiNaC::power::expand_mul (
    const mul & m,
    const numeric & n,
    unsigned options,
    bool from_expand = false) [static], [protected]
```

Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::expand\\_options::expand\\_rename\\_idx](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::get\\_all\\_dummy\\_indices\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::mul::info\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [mul](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [eval\(\)](#), [expand\(\)](#), and [expand\\_add\\_2\(\)](#).

## 8.123.4 Friends And Related Symbol Documentation

### 8.123.4.1 mul

```
friend class mul [friend]
```

References [mul](#).

Referenced by [eval\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [expand\\_mul\(\)](#), and [mul](#).

## 8.123.5 Member Data Documentation

### 8.123.5.1 basis

```
ex GiNaC::power::basis [protected]
```

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [power\(\)](#), [power\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 8.123.5.2 exponent

```
ex GiNaC::power::exponent [protected]
```

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [power\(\)](#), [power\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

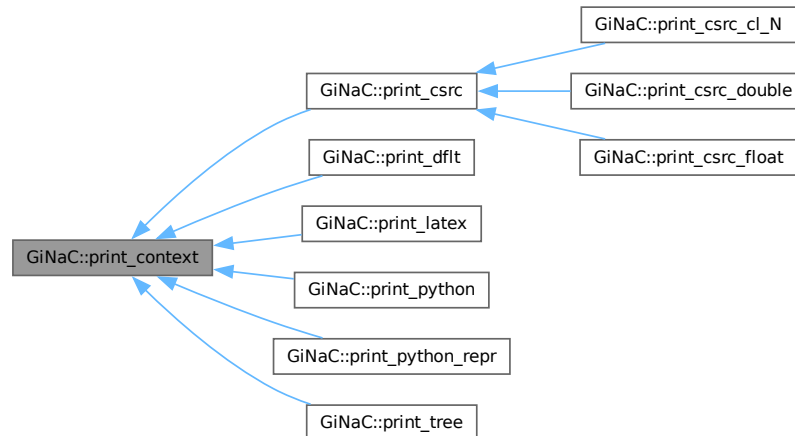
- [power.h](#)
- [normal.cpp](#)
- [power.cpp](#)
- [pseries.cpp](#)

## 8.124 GiNaC::print\_context Class Reference

Base class for print\_contexts.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_context:



### Public Member Functions

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Public Attributes

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 8.124.1 Detailed Description

Base class for print\_contexts.

### 8.124.2 Constructor & Destructor Documentation

#### 8.124.2.1 print\_context()

```
GiNaC::print_context::print_context (
    std::ostream & os,
    unsigned options = 0)
```

References [options](#), [print\\_context\(\)](#), and [s](#).

Referenced by [print\\_context\(\)](#), [GiNaC::print\\_csrf::print\\_csrf\(\)](#), [GiNaC::print\\_dflt::print\\_dflt\(\)](#), [GiNaC::print\\_latex::print\\_latex\(\)](#), [GiNaC::print\\_python::print\\_python\(\)](#), [GiNaC::print\\_python\\_repr::print\\_python\\_repr\(\)](#), [GiNaC::print\\_tree::print\\_tree\(\)](#), and [GiNaC::print\\_tree::print\\_tree\(\)](#).

### 8.124.2.2 ~print\_context()

```
virtual GiNaC::print_context::~~print_context () [inline], [virtual]
```

## 8.124.3 Member Data Documentation

### 8.124.3.1 s

```
std::ostream& GiNaC::print_context::s
```

stream to output to

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::basic::do\\_print\(\)](#), [GiNaC::basic\\_log\\_kernel::do\\_print\(\)](#), [GiNaC::constant::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::expairseq::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::Gt::do\\_print\(\)](#), [GiNaC::idx::do\\_print\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::integration\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::matrix::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [GiNaC::symbol::do\\_print\(\)](#), [GiNaC::symmetry::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [GiNaC::wildcard::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::ncmul::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [GiNaC::constant::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::Gt::do\\_print\\_latex\(\)](#), [GiNaC::idx::do\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::matrix::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::spinidx::do\\_print\\_latex\(\)](#), [GiNaC::symbol::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_python\\_repr\(\)](#), [GiNaC::constant::do\\_print\\_python\\_repr\(\)](#), [GiNaC::container< C >::do\\_print\\_python\\_repr\(\)](#), [GiNaC::matrix::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::numeric::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::symbol::do\\_print\\_python\\_repr\(\)](#), [GiNaC::wildcard::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [print\\_context\(\)](#), [GiNaC::matrix::print\\_elements\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::indexed::print\\_indexed\(\)](#), [GiNaC::print\\_integer\\_csrc\(\)](#), [GiNaC::numeric::print\\_numeric\(\)](#), [GiNaC::print\\_operator\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

### 8.124.3.2 options

```
unsigned GiNaC::print_context::options
```

option flags

Referenced by [GiNaC::get\\_print\\_options\(\)](#), [print\\_context\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), and [GiNaC::set\\_print\\_options\(\)](#).

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 8.125 GiNaC::print\_context\_options Class Reference

This class stores information about a registered [print\\_context](#) class.

```
#include <print.h>
```

### Public Member Functions

- [print\\_context\\_options](#) (const char \*n, const char \*p, unsigned i)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- unsigned [get\\_id](#) () const

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- unsigned [id](#)  
*ID number (assigned automatically).*

### 8.125.1 Detailed Description

This class stores information about a registered [print\\_context](#) class.

### 8.125.2 Constructor & Destructor Documentation

#### 8.125.2.1 print\_context\_options()

```
GiNaC::print_context_options::print_context_options (  
    const char * n,  
    const char * p,  
    unsigned i) [inline]
```

References [id](#), [name](#), and [parent\\_name](#).

### 8.125.3 Member Function Documentation

#### 8.125.3.1 get\_name()

```
const char * GiNaC::print_context_options::get_name () const [inline]
```

References [name](#).

### 8.125.3.2 `get_parent_name()`

```
const char * GiNaC::print_context_options::get_parent_name () const [inline]
```

References [parent\\_name](#).

### 8.125.3.3 `get_id()`

```
unsigned GiNaC::print_context_options::get_id () const [inline]
```

References [id](#).

Referenced by [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

## 8.125.4 Member Data Documentation

### 8.125.4.1 `name`

```
const char* GiNaC::print_context_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#), and [print\\_context\\_options\(\)](#).

### 8.125.4.2 `parent_name`

```
const char* GiNaC::print_context_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#), and [print\\_context\\_options\(\)](#).

### 8.125.4.3 `id`

```
unsigned GiNaC::print_context_options::id [private]
```

ID number (assigned automatically).

Referenced by [get\\_id\(\)](#), and [print\\_context\\_options\(\)](#).

The documentation for this class was generated from the following file:

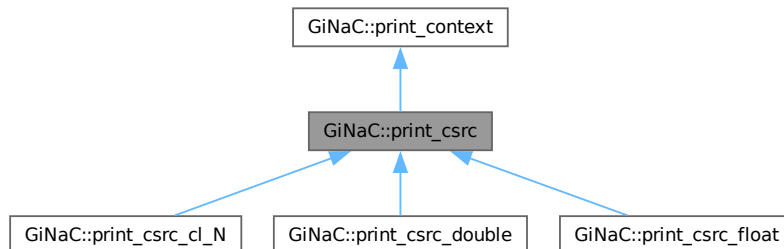
- [print.h](#)

## 8.126 GiNaC::print\_csrc Class Reference

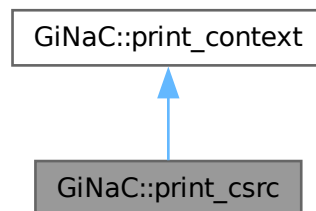
Base context for C source output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc:



Collaboration diagram for GiNaC::print\_csrc:



### Public Member Functions

- `print_csrc` (std::ostream &, unsigned `options`=0)

### Public Member Functions inherited from `GiNaC::print_context`

- `print_context` (std::ostream &, unsigned `options`=0)
- virtual `~print_context` ()

### Additional Inherited Members

### Public Attributes inherited from `GiNaC::print_context`

- std::ostream & `s`  
*stream to output to*
- unsigned `options`  
*option flags*

### 8.126.1 Detailed Description

Base context for C source output.

### 8.126.2 Constructor & Destructor Documentation

#### 8.126.2.1 `print_csrc()`

```
GiNaC::print_csrc::print_csrc (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_context::print\\_context\(\)](#).

Referenced by [GiNaC::print\\_csrc\\_cl\\_N::print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::print\\_csrc\\_double::print\\_csrc\\_double\(\)](#), and [GiNaC::print\\_csrc\\_float::print\\_csrc\\_float\(\)](#).

The documentation for this class was generated from the following files:

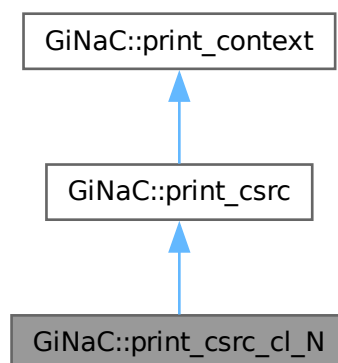
- [print.h](#)
- [print.cpp](#)

## 8.127 GiNaC::print\_csrc\_cl\_N Class Reference

Context for C source output using CLN numbers.

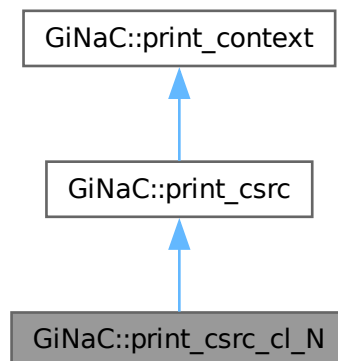
```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_cl\_N:





Collaboration diagram for GiNaC::print\_csrc\_cl\_N:



### Public Member Functions

- [print\\_csrc\\_cl\\_N](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 8.127.1 Detailed Description

Context for C source output using CLN numbers.

## 8.127.2 Constructor & Destructor Documentation

### 8.127.2.1 `print_csrc_cl_N()`

```
GiNaC::print_csrc_cl_N::print_csrc_cl_N (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_csrc::print\\_csrc\(\)](#).

The documentation for this class was generated from the following files:

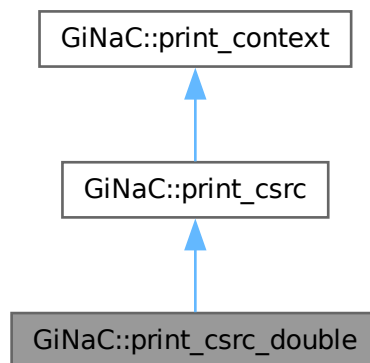
- [print.h](#)
- [print.cpp](#)

## 8.128 GiNaC::print\_csrc\_double Class Reference

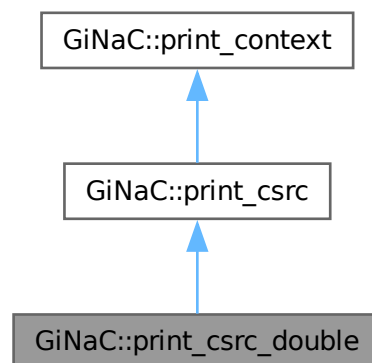
Context for C source output using double precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_double:



Collaboration diagram for GiNaC::print\_csrc\_double:



### Public Member Functions

- [print\\_csrc\\_double](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 8.128.1 Detailed Description

Context for C source output using double precision.

## 8.128.2 Constructor & Destructor Documentation

### 8.128.2.1 `print_csrc_double()`

```
GiNaC::print_csrc_double::print_csrc_double (
    std::ostream & os,
    unsigned options = 0)
```

References [GiNaC::print\\_csrc::print\\_csrc\(\)](#).

The documentation for this class was generated from the following files:

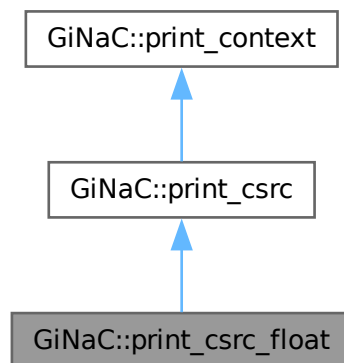
- [print.h](#)
- [print.cpp](#)

## 8.129 GiNaC::print\_csrc\_float Class Reference

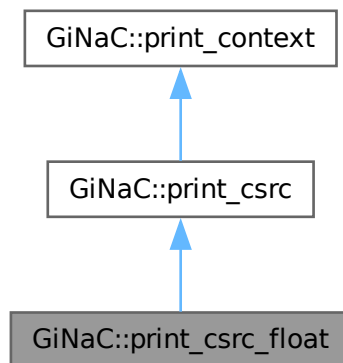
Context for C source output using float precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_float:



Collaboration diagram for GiNaC::print\_csrc\_float:



#### Public Member Functions

- [print\\_csrc\\_float](#) (std::ostream &, unsigned [options](#)=0)

#### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

#### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

#### Additional Inherited Members

#### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 8.129.1 Detailed Description

Context for C source output using float precision.

## 8.129.2 Constructor & Destructor Documentation

### 8.129.2.1 `print_csrc_float()`

```
GiNaC::print_csrc_float::print_csrc_float (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_csrc::print\\_csrc\(\)](#).

The documentation for this class was generated from the following files:

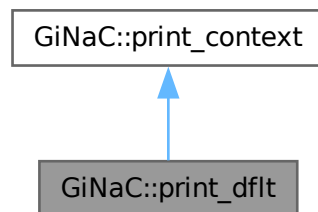
- [print.h](#)
- [print.cpp](#)

## 8.130 GiNaC::print\_dflt Class Reference

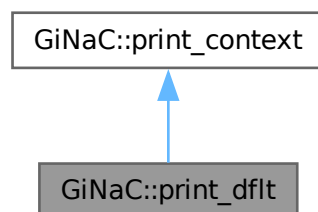
Context for default (ginsh-parsable) output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_dflt:



Collaboration diagram for GiNaC::print\_dflt:



## Public Member Functions

- [print\\_dflt](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 8.130.1 Detailed Description

Context for default (ginsh-parsable) output.

### 8.130.2 Constructor & Destructor Documentation

#### 8.130.2.1 [print\\_dflt\(\)](#)

```
GiNaC::print_dflt::print_dflt (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_context::print\\_context\(\)](#).

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 8.131 GiNaC::print\_functor Class Reference

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

```
#include <print.h>
```

## Public Member Functions

- [print\\_functor](#) ()
- [print\\_functor](#) (const [print\\_functor](#) &other)
- [print\\_functor](#) (std::unique\_ptr< [print\\_functor\\_impl](#) > impl\_)
- template<class T, class C>  
  [print\\_functor](#) (void f(const T &, const C &, unsigned))
- template<class T, class C>  
  [print\\_functor](#) (void(T::\*f)(const C &, unsigned) const)
- [print\\_functor](#) & [operator=](#) (const [print\\_functor](#) &other)
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const
- bool [is\\_valid](#) () const

## Private Attributes

- std::unique\_ptr< [print\\_functor\\_impl](#) > [impl](#)

### 8.131.1 Detailed Description

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

Its main purpose is to hide the difference between member functions and nonmember functions behind one unified [operator\(\)](#) interface. [print\\_functor](#) has value semantics and acts as a smart pointer (with deep copy) to a class derived from [print\\_functor\\_impl](#) which implements the actual function call.

### 8.131.2 Constructor & Destructor Documentation

#### 8.131.2.1 [print\\_functor](#)() [1/5]

```
GiNaC::print_functor::print_functor () [inline]
```

References [impl](#).

Referenced by [operator=\(\)](#), and [print\\_functor\(\)](#).

#### 8.131.2.2 [print\\_functor](#)() [2/5]

```
GiNaC::print_functor::print_functor (
    const print_functor & other) [inline]
```

References [impl](#), and [print\\_functor\(\)](#).

#### 8.131.2.3 [print\\_functor](#)() [3/5]

```
GiNaC::print_functor::print_functor (
    std::unique_ptr< print\_functor\_impl > impl_) [inline]
```

References [impl](#).



**8.131.2.4 print\_functor()** [4/5]

```
template<class T, class C>
GiNaC::print_functor::print_functor (
    void fconst T &, const C &, unsigned) [inline]
```

References [impl](#).

**8.131.2.5 print\_functor()** [5/5]

```
template<class T, class C>
GiNaC::print_functor::print_functor (
    void(T::* f)(const C &, unsigned) const) [inline]
```

References [impl](#).

**8.131.3 Member Function Documentation****8.131.3.1 operator=()**

```
print\_functor & GiNaC::print_functor::operator= (
    const print\_functor & other) [inline]
```

References [impl](#), and [print\\_functor\(\)](#).

**8.131.3.2 operator>()()**

```
void GiNaC::print_functor::operator() (
    const basic & obj,
    const print\_context & c,
    unsigned level) const [inline]
```

**8.131.3.3 is\_valid()**

```
bool GiNaC::print_functor::is_valid () const [inline]
```

References [impl](#).

**8.131.4 Member Data Documentation****8.131.4.1 impl**

```
std::unique_ptr<print\_functor\_impl> GiNaC::print_functor::impl [private]
```

Referenced by [is\\_valid\(\)](#), [operator=\(\)](#), [print\\_functor\(\)](#), [print\\_functor\(\)](#), [print\\_functor\(\)](#), [print\\_functor\(\)](#), and [print\\_functor\(\)](#).

The documentation for this class was generated from the following file:

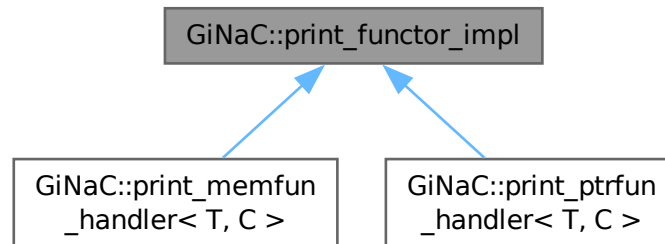
- [print.h](#)

## 8.132 GiNaC::print\_functor\_impl Class Reference

Base class for [print\\_functor](#) handlers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_functor\_impl:



### Public Member Functions

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

### 8.132.1 Detailed Description

Base class for [print\\_functor](#) handlers.

### 8.132.2 Constructor & Destructor Documentation

#### 8.132.2.1 ~print\_functor\_impl()

```
virtual GiNaC::print_functor_impl::~~print_functor_impl () [inline], [virtual]
```

### 8.132.3 Member Function Documentation

#### 8.132.3.1 duplicate()

```
virtual print\_functor\_impl * GiNaC::print_functor_impl::duplicate () const [pure virtual]
```

Implemented in [GiNaC::print\\_memfun\\_handler< T, C >](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >](#).

### 8.132.3.2 operator>()

```
virtual void GiNaC::print_functor_impl::operator() (  
    const basic & obj,  
    const print_context & c,  
    unsigned level) const [pure virtual]
```

Implemented in [GiNaC::print\\_memfun\\_handler< T, C >](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >](#).

The documentation for this class was generated from the following file:

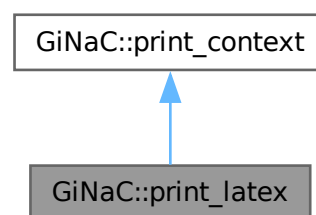
- [print.h](#)

## 8.133 GiNaC::print\_latex Class Reference

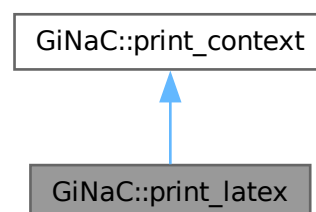
Context for latex-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_latex:



Collaboration diagram for GiNaC::print\_latex:



## Public Member Functions

- [print\\_latex](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 8.133.1 Detailed Description

Context for latex-parsable output.

### 8.133.2 Constructor & Destructor Documentation

#### 8.133.2.1 [print\\_latex\(\)](#)

```
GiNaC::print_latex::print_latex (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_context::print\\_context\(\)](#).

The documentation for this class was generated from the following files:

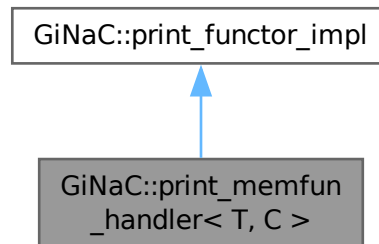
- [print.h](#)
- [print.cpp](#)

## 8.134 GiNaC::print\_memfun\_handler< T, C > Class Template Reference

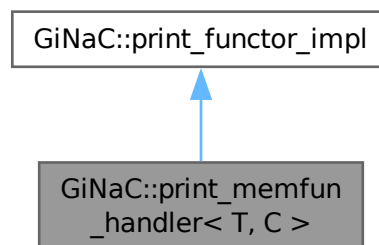
[print\\_functor](#) handler for member functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_memfun\_handler< T, C >:



Collaboration diagram for GiNaC::print\_memfun\_handler< T, C >:



### Public Types

- typedef void(T::\* [F](#)) (const C &c, unsigned level) const

### Public Member Functions

- [print\\_memfun\\_handler](#) ([F](#) f\_)
- [print\\_memfun\\_handler](#) \* [duplicate](#) () const override
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override

## Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()

## Private Attributes

- [F f](#)

### 8.134.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_memfun_handler< T, C >
```

[print\\_functor](#) handler for member functions of class T, context type C

### 8.134.2 Member Typedef Documentation

#### 8.134.2.1 F

```
template<class T, class C>
typedef void(T::* GiNaC::print\_memfun\_handler< T, C >::F) (const C &c, unsigned level) const
```

### 8.134.3 Constructor & Destructor Documentation

#### 8.134.3.1 print\_memfun\_handler()

```
template<class T, class C>
GiNaC::print\_memfun\_handler< T, C >::print_memfun_handler (
    F f_) [inline]
```

References [f](#).

Referenced by [duplicate\(\)](#).

### 8.134.4 Member Function Documentation

#### 8.134.4.1 duplicate()

```
template<class T, class C>
print\_memfun\_handler * GiNaC::print\_memfun\_handler< T, C >::duplicate () const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [print\\_memfun\\_handler\(\)](#).

#### 8.134.4.2 operator>()()

```
template<class T, class C>
void GiNaC::print_memfun_handler< T, C >::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [f](#).

### 8.134.5 Member Data Documentation

#### 8.134.5.1 f

```
template<class T, class C>
F GiNaC::print_memfun_handler< T, C >::f [private]
```

Referenced by [operator>\(\)\(\)](#), and [print\\_memfun\\_handler\(\)](#).

The documentation for this class was generated from the following file:

- [print.h](#)

## 8.135 GiNaC::print\_options Class Reference

Flags to control the behavior of a [print\\_context](#).

```
#include <print.h>
```

### Public Types

- enum { [print\\_index\\_dimensions](#) = 0x0001 }

### 8.135.1 Detailed Description

Flags to control the behavior of a [print\\_context](#).

### 8.135.2 Member Enumeration Documentation

#### 8.135.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<a href="#">print_index_dimensions</a>	print the dimensions of indices
--	---------------------------------

The documentation for this class was generated from the following file:

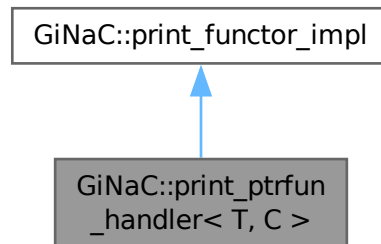
- [print.h](#)

## 8.136 GiNaC::print\_ptrfun\_handler< T, C > Class Template Reference

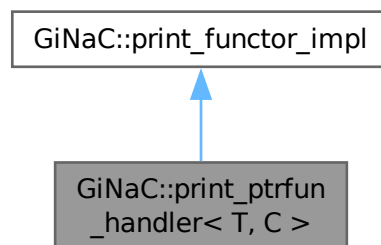
[print\\_functor](#) handler for pointer-to-functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_ptrfun\_handler< T, C >:



Collaboration diagram for GiNaC::print\_ptrfun\_handler< T, C >:



### Public Types

- typedef void(\* [F](#)) (const T &, const C &, unsigned)

### Public Member Functions

- [print\\_ptrfun\\_handler](#) (F f\_)
- [print\\_ptrfun\\_handler](#) \* [duplicate](#) () const override
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override



## Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()

## Private Attributes

- [F f](#)

### 8.136.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_ptrfun_handler< T, C >
```

[print\\_functor](#) handler for pointer-to-functions of class T, context type C

### 8.136.2 Member Typedef Documentation

#### 8.136.2.1 F

```
template<class T, class C>
typedef void(* GiNaC::print\_ptrfun\_handler< T, C >::F) (const T &, const C &, unsigned)
```

### 8.136.3 Constructor & Destructor Documentation

#### 8.136.3.1 print\_ptrfun\_handler()

```
template<class T, class C>
GiNaC::print\_ptrfun\_handler< T, C >::print_ptrfun_handler (
    F f_) [inline]
```

References [f](#).

Referenced by [duplicate\(\)](#).

### 8.136.4 Member Function Documentation

#### 8.136.4.1 duplicate()

```
template<class T, class C>
print\_ptrfun\_handler * GiNaC::print\_ptrfun\_handler< T, C >::duplicate () const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [print\\_ptrfun\\_handler\(\)](#).

#### 8.136.4.2 `operator>()()`

```
template<class T, class C>
void GiNaC::print_ptrfun_handler< T, C >::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [f](#).

### 8.136.5 Member Data Documentation

#### 8.136.5.1 `f`

```
template<class T, class C>
F GiNaC::print_ptrfun_handler< T, C >::f [private]
```

Referenced by [operator>\(\)\(\)](#), and [print\\_ptrfun\\_handler\(\)](#).

The documentation for this class was generated from the following file:

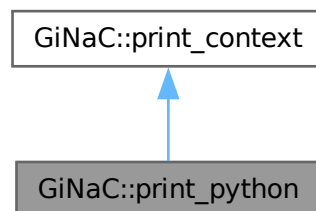
- [print.h](#)

### 8.137 GiNaC::print\_python Class Reference

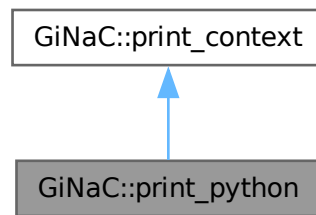
Context for python pretty-print output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python:



Collaboration diagram for GiNaC::print\_python:



### Public Member Functions

- [print\\_python](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 8.137.1 Detailed Description

Context for python pretty-print output.

## 8.137.2 Constructor & Destructor Documentation

### 8.137.2.1 print\_python()

```

GiNaC::print_python::print_python (
    std::ostream & os,
    unsigned options = 0)

```

References [GiNaC::print\\_context::print\\_context\(\)](#).

The documentation for this class was generated from the following files:

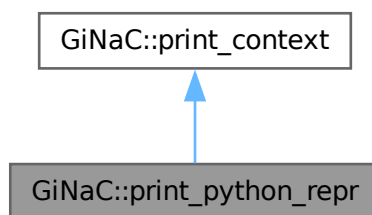
- [print.h](#)
- [print.cpp](#)

## 8.138 GiNaC::print\_python\_repr Class Reference

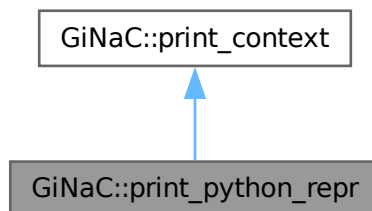
Context for python-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python\_repr:



Collaboration diagram for GiNaC::print\_python\_repr:



### Public Member Functions

- [print\\_python\\_repr](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- `std::ostream & s`  
*stream to output to*
- `unsigned options`  
*option flags*

### 8.138.1 Detailed Description

Context for python-parsable output.

### 8.138.2 Constructor & Destructor Documentation

#### 8.138.2.1 `print_python_repr()`

```
GiNaC::print_python_repr::print_python_repr (  
    std::ostream & os,  
    unsigned options = 0)
```

References [GiNaC::print\\_context::print\\_context\(\)](#).

The documentation for this class was generated from the following files:

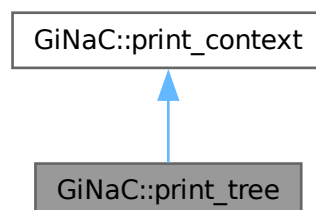
- [print.h](#)
- [print.cpp](#)

## 8.139 GiNaC::print\_tree Class Reference

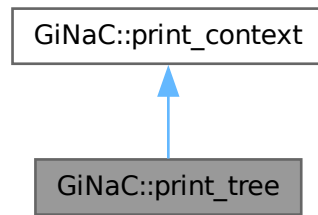
Context for tree-like output for debugging.

```
#include <print.h>
```

Inheritance diagram for `GiNaC::print_tree`:



Collaboration diagram for `GiNaC::print_tree`:



### Public Member Functions

- [print\\_tree](#) (unsigned d)
- [print\\_tree](#) (std::ostream &, unsigned options=0, unsigned d=4)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned options=0)
- virtual [~print\\_context](#) ()

### Public Attributes

- const unsigned [delta\\_indent](#)  
*size of indentation step*

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 8.139.1 Detailed Description

Context for tree-like output for debugging.

## 8.139.2 Constructor & Destructor Documentation

### 8.139.2.1 `print_tree()` [1/2]

```
GiNaC::print_tree::print_tree (
    unsigned d)
```

References [delta\\_indent](#), and [GiNaC::print\\_context::print\\_context\(\)](#).

### 8.139.2.2 print\_tree() [2/2]

```
GiNaC::print_tree::print_tree (
    std::ostream & os,
    unsigned options = 0,
    unsigned d = 4)
```

References [delta\\_indent](#), and [GiNaC::print\\_context::print\\_context\(\)](#).

## 8.139.3 Member Data Documentation

### 8.139.3.1 delta\_indent

```
const unsigned GiNaC::print_tree::delta_indent
```

size of indentation step

Referenced by [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::expair::print\(\)](#), [print\\_tree\(\)](#), and [print\\_tree\(\)](#).

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 8.140 GiNaC::archive\_node::property Struct Reference

Archived property (data type, name and associated data).

```
#include <archive.h>
```

### Public Member Functions

- [property](#) ()
- [property](#) ([archive\\_atom](#) n, [property\\_type](#) t, unsigned v)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- [archive\\_atom](#) name  
*Name of property.*
- unsigned [value](#)  
*Stored value.*

### 8.140.1 Detailed Description

Archived property (data type, name and associated data).

### 8.140.2 Constructor & Destructor Documentation

#### 8.140.2.1 `property()` [1/2]

```
GiNaC::archive_node::property::property () [inline]
```

#### 8.140.2.2 `property()` [2/2]

```
GiNaC::archive_node::property::property (  
    archive_atom n,  
    property_type t,  
    unsigned v) [inline]
```

References [name](#), [type](#), and [value](#).

### 8.140.3 Member Data Documentation

#### 8.140.3.1 `type`

```
property_type GiNaC::archive_node::property::type
```

Data type of property.

Referenced by [property\(\)](#).

#### 8.140.3.2 `name`

```
archive_atom GiNaC::archive_node::property::name
```

Name of property.

Referenced by [property\(\)](#).

#### 8.140.3.3 `value`

```
unsigned GiNaC::archive_node::property::value
```

Stored value.

Referenced by [property\(\)](#).

The documentation for this struct was generated from the following file:

- [archive.h](#)



## 8.141 GiNaC::archive\_node::property\_info Struct Reference

Information about a stored property.

```
#include <archive.h>
```

### Public Member Functions

- [property\\_info](#) ()
- [property\\_info](#) ([property\\_type](#) t, const std::string &n, unsigned c=1)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- std::string [name](#)  
*Name of property.*
- unsigned [count](#)  
*Number of occurrences.*

### 8.141.1 Detailed Description

Information about a stored property.

A vector of these structures is returned by [get\\_properties\(\)](#).

See also

[get\\_properties](#)

### 8.141.2 Constructor & Destructor Documentation

#### 8.141.2.1 property\_info() [1/2]

```
GiNaC::archive_node::property_info::property_info () [inline]
```

#### 8.141.2.2 property\_info() [2/2]

```
GiNaC::archive_node::property_info::property_info (  
    property\_type t,  
    const std::string & n,  
    unsigned c = 1) [inline]
```

References [count](#), [name](#), and [type](#).

### 8.141.3 Member Data Documentation

#### 8.141.3.1 type

`property_type` `GiNaC::archive_node::property_info::type`

Data type of property.

Referenced by [property\\_info\(\)](#).

#### 8.141.3.2 name

`std::string` `GiNaC::archive_node::property_info::name`

Name of property.

Referenced by [property\\_info\(\)](#).

#### 8.141.3.3 count

`unsigned` `GiNaC::archive_node::property_info::count`

Number of occurrences.

Referenced by [property\\_info\(\)](#).

The documentation for this struct was generated from the following file:

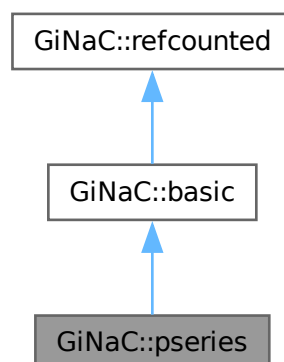
- [archive.h](#)

## 8.142 GiNaC::pseries Class Reference

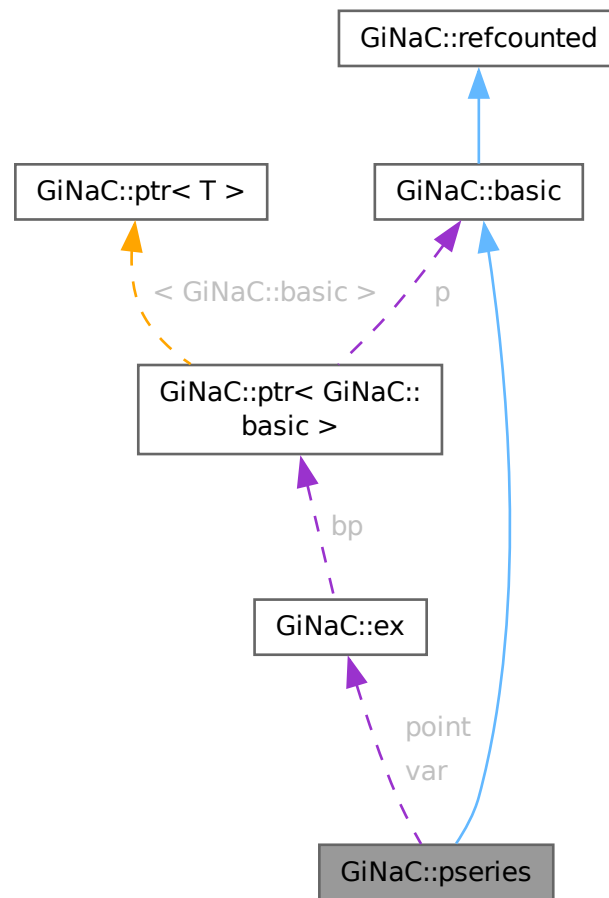
This class holds a extended truncated power series (positive and negative integer powers).

```
#include <pseries.h>
```

Inheritance diagram for `GiNaC::pseries`:



Collaboration diagram for GiNaC::pseries:



## Public Member Functions

- `pseries` (const `ex` &rel\_, const `epvector` &ops\_)  
Construct *pseries* from a vector of coefficients and powers.
- `pseries` (const `ex` &rel\_, `epvector` &&ops\_)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- `size_t nops` () const override  
Return the number of operands including a possible order term.
- `ex op` (size\_t i) const override  
Return the *i*th term in the series when represented as a sum.
- int `degree` (const `ex` &s) const override  
Return degree of highest power of the series.
- int `ldegree` (const `ex` &s) const override  
Return degree of lowest power of the series.
- `ex coeff` (const `ex` &s, int n=1) const override

- Return coefficient of degree  $n$  in power series if  $s$  is the expansion variable.*

  - `ex collect` (const `ex` & $s$ , bool distributed=false) const override

*Does nothing.*
- `ex eval` () const override

*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex evalf` () const override

*Evaluate coefficients numerically.*
- `ex series` (const `relational` & $r$ , int order, unsigned options=0) const override

*Re-expansion of a `pseries` object.*
- `ex subs` (const `exmap` & $m$ , unsigned options=0) const override

*Substitute a set of objects by arbitrary expressions.*
- `ex normal` (`exmap` & $repl$ , `exmap` & $rev\_lookup$ , `lst` & $modifier$ ) const override

*Implementation of `ex::normal()` for `pseries`.*
- `ex expand` (unsigned options=0) const override

*Implementation of `ex::expand()` for a power series.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- `ex eval_integ` () const override

*Evaluate integrals, if result is known.*
- `ex evalm` () const override

*Evaluate sums, products and integer powers of matrices.*
- void `archive` (`archive_node` & $n$ ) const override

*Save (a.k.a.*
- void `read_archive` (const `archive_node` & $n$ , `lst` & $syms$ ) override

*Read (a.k.a.*
- `ex get_var` () const

*Get the expansion variable.*
- `ex get_point` () const

*Get the expansion point.*
- `ex convert_to_poly` (bool no\_order=false) const

*Convert the `pseries` object to an ordinary polynomial.*
- bool `is_compatible_to` (const `pseries` & $other$ ) const

*Check whether series is compatible to another series (expansion variable and point are the same.*
- bool `is_zero` () const

*Check whether series has the value zero.*
- bool `is_terminating` () const

*Returns true if there is no order term, i.e.*
- `ex coeffop` (size\_t  $i$ ) const

*Get coefficients and exponents.*
- `ex exponop` (size\_t  $i$ ) const
- `ex add_series` (const `pseries` & $other$ ) const

*Add one series object to another, producing a `pseries` object that represents the sum.*
- `ex mul_const` (const `numeric` & $other$ ) const

*Multiply a `pseries` object with a numeric constant, producing a `pseries` object that represents the product.*
- `ex mul_series` (const `pseries` & $other$ ) const

*Multiply one `pseries` object to another, producing a `pseries` object that represents the product.*
- `ex power_const` (const `numeric` & $p$ , int deg) const

*Compute the  $p$ -th power of a series.*
- `pseries shift_exponents` (int deg) const

*Return a new `pseries` object with the powers shifted by  $deg$ .*

Public Member Functions inherited from **GiNaC::basic**

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level=0`) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual bool `info` (unsigned `inf`) const  
*Information about the object.*
- virtual `ex operator[]` (const `ex` &`index`) const
- virtual `ex operator[]` (size\_t `i`) const
- virtual `ex` & `let_op` (size\_t `i`)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &`index`)
- virtual `ex` & `operator[]` (size\_t `i`)
- virtual bool `has` (const `ex` &`other`, unsigned `options=0`) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &`pattern`, `exmap` &`repls`) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &`f`) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &`v`) const
- virtual bool `is_polynomial` (const `ex` &`var`) const  
*Check whether this is a polynomial in the given variables.*
- virtual `ex to_rational` (`exmap` &`repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &`repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &`xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &`self`, const `ex` &`other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const

- `template<class T>`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

## Protected Member Functions

- `ex derivative (const symbol &s) const override`  
*Implementation of `ex::diff()` for a power series.*
- `void print_series (const print_context &c, const char *openbrace, const char *closebrace, const char *mul←  
_sym, const char *pow_sym, unsigned level) const`
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_latex (const print_latex &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`
- `void do_print_python (const print_python &c, unsigned level) const`
- `void do_print_python_repr (const print_python_repr &c, unsigned level) const`

## Protected Member Functions inherited from `GiNaC::basic`

- `basic ()`
- `virtual ex eval_ncmul (const exvector &v) const`
- `virtual bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type (const basic &other) const`

- Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [epvector](#) [seq](#)  
*Vector of {coefficient, power} pairs.*
- [ex](#) [var](#)  
*Series variable (holds a symbol).*
- [ex](#) [point](#)  
*Expansion point.*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.142.1 Detailed Description

This class holds a extended truncated power series (positive and negative integer powers).

It consists of expression coefficients (only non-zero coefficients are stored), an expansion variable and an expansion point. Other classes must provide members to convert into this type.

## 8.142.2 Constructor & Destructor Documentation

### 8.142.2.1 [pseries\(\)](#) [1/2]

```
GiNaC::pseries::pseries (
    const ex & rel\_,
    const epvector & ops\_)
```

Construct pseries from a vector of coefficients and powers.

[expair.rest](#) holds the coefficient, [expair.coef](#) holds the power. The powers must be integers (positive or negative) and in ascending order; the last coefficient can be `Order(_ex1)` to represent a truncated, non-terminating series.

#### Parameters

<a href="#">rel</a> ↔	expansion variable and point (must hold a relational)
—	

<code>ops</code> ↔	vector of {coefficient, power} pairs (coefficient must not be zero)
—	

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

Referenced by [add\\_series\(\)](#), [derivative\(\)](#), [is\\_compatible\\_to\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [power\\_const\(\)](#), [series\(\)](#), and [shift\\_exponents\(\)](#).

### 8.142.2.2 pseries() [2/2]

```
GiNaC::pseries::pseries (
    const ex & rel\_,
    epvector && ops\_)
```

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

## 8.142.3 Member Function Documentation

### 8.142.3.1 precedence()

```
unsigned GiNaC::pseries::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_series\(\)](#).

### 8.142.3.2 nops()

```
size_t GiNaC::pseries::nops () const [override], [virtual]
```

Return the number of operands including a possible order term.

Reimplemented from [GiNaC::basic](#).

References [seq](#).

Referenced by [coeffop\(\)](#), [exponop\(\)](#), and [GiNaC::log\\_series\(\)](#).

### 8.142.3.3 op()

```
ex GiNaC::pseries::op (
    size_t i) const [override], [virtual]
```

Return the ith term in the series when represented as a sum.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).



### 8.142.3.4 degree()

```
int GiNaC::pseries::degree (
    const ex & s) const [override], [virtual]
```

Return degree of highest power of the series.

This is usually the exponent of the Order term. If s is not the expansion variable of the series, the series is examined termwise.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [seq](#), and [var](#).

Referenced by [mul\\_series\(\)](#), [power\\_const\(\)](#), and [series\(\)](#).

### 8.142.3.5 ldegree()

```
int GiNaC::pseries::ldegree (
    const ex & s) const [override], [virtual]
```

Return degree of lowest power of the series.

This is usually the exponent of the leading term. If s is not the expansion variable of the series, the series is examined termwise. If s is the expansion variable but the expansion point is not zero the series is not expanded to find the degree. I.e.:  $(1-x) + (1-x)^2 + \text{Order}((1-x)^3)$  has `ldegree(x)` 1, not 0.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), and [power\\_const\(\)](#).

### 8.142.3.6 coeff()

```
ex GiNaC::pseries::coeff (
    const ex & s,
    int n = 1) const [override], [virtual]
```

Return coefficient of degree n in power series if s is the expansion variable.

If the expansion point is nonzero, by definition the n=1 coefficient in s of  $a+b*(s-z)+c*(s-z)^2+\text{Order}((s-z)^3)$  is b (assuming the expansion took place in the s in the first place). If s is not the expansion variable, an attempt is made to convert the series to a polynomial and return the corresponding coefficient from there.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [convert\\_to\\_poly\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), [op\(\)](#), [power\\_const\(\)](#), and [read\\_archive\(\)](#).

**8.142.3.7 collect()**

```
ex GiNaC::pseries::collect (
    const ex & s,
    bool distributed = false) const [override], [virtual]
```

Does nothing.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.142.3.8 eval()**

```
ex GiNaC::pseries::eval () const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::basic::hold\(\)](#).

**8.142.3.9 evalf()**

```
ex GiNaC::pseries::evalf () const [override], [virtual]
```

Evaluate coefficients numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [point](#), [seq](#), and [var](#).

**8.142.3.10 series()**

```
ex GiNaC::pseries::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Re-expansion of a pseries object.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex1](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::relational::lhs\(\)](#), [point](#), [pseries\(\)](#), [GiNaC::relational::rhs\(\)](#), [seq](#), [GiNaC::ex::series\(\)](#), and [var](#).

**8.142.3.11 subs()**

```
ex GiNaC::pseries::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [convert\\_to\\_poly\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [point](#), [seq](#), [GiNaC::ex::subs\(\)](#), and [var](#).

**8.142.3.12 normal()**

```
ex GiNaC::pseries::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for pseries.

It normalizes each coefficient and replaces the series by a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [pseries\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [seq](#), and [var](#).

**8.142.3.13 expand()**

```
ex GiNaC::pseries::expand (
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::expand\(\)](#) for a power series.

It expands all the terms individually and returns the resulting series as a new pseries.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).

#### 8.142.3.14 conjugate()

```
ex GiNaC::pseries::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

#### 8.142.3.15 real\_part()

```
ex GiNaC::pseries::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

#### 8.142.3.16 imag\_part()

```
ex GiNaC::pseries::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

#### 8.142.3.17 eval\_integ()

```
ex GiNaC::pseries::eval_integ () const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [point](#), [seq](#), and [var](#).

#### 8.142.3.18 evalm()

```
ex GiNaC::pseries::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).

### 8.142.3.19 archive()

```
void GiNaC::pseries::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [point](#), [seq](#), and [var](#).

### 8.142.3.20 read\_archive()

```
void GiNaC::pseries::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_by\\_loc\(\)](#), [GiNaC::archive\\_node::find\\_property\\_range\(\)](#), [point](#), [seq](#), and [var](#).

### 8.142.3.21 derivative()

```
ex GiNaC::pseries::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power series.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

### 8.142.3.22 get\_var()

```
ex GiNaC::pseries::get_var () const [inline]
```

Get the expansion variable.

References [GiNaC::basic::ex](#), and [var](#).

**8.142.3.23 get\_point()**

```
ex GiNaC::pseries::get_point () const [inline]
```

Get the expansion point.

References [GiNaC::basic::ex](#), and [point](#).

**8.142.3.24 convert\_to\_poly()**

```
ex GiNaC::pseries::convert_to_poly (
    bool no_order = false) const
```

Convert the pseries object to an ordinary polynomial.

**Parameters**

<i>no_order</i>	flag: discard higher order terms
-----------------	----------------------------------

References [GiNaC::basic::ex](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [series\(\)](#), and [subs\(\)](#).

**8.142.3.25 is\_compatible\_to()**

```
bool GiNaC::pseries::is_compatible_to (
    const pseries & other) const [inline]
```

Check whether series is compatible to another series (expansion variable and point are the same).

References [point](#), [pseries\(\)](#), and [var](#).

Referenced by [add\\_series\(\)](#), and [mul\\_series\(\)](#).

**8.142.3.26 is\_zero()**

```
bool GiNaC::pseries::is_zero () const [inline]
```

Check whether series has the value zero.

References [seq](#).

Referenced by [power\\_const\(\)](#).

**8.142.3.27 is\_terminating()**

```
bool GiNaC::pseries::is_terminating () const
```

Returns true if there is no order term, i.e.

the series terminates and false otherwise.

References [GiNaC::is\\_order\\_function\(\)](#), and [seq](#).

Referenced by [GiNaC::is\\_terminating\(\)](#), and [GiNaC::log\\_series\(\)](#).

**8.142.3.28 coeffop()**

```
ex GiNaC::pseries::coeffop (
    size_t i) const
```

Get coefficients and exponents.

References [GiNaC::basic::ex](#), [nops\(\)](#), and [seq](#).

**8.142.3.29 exponop()**

```
ex GiNaC::pseries::exponop (
    size_t i) const
```

References [GiNaC::basic::ex](#), [nops\(\)](#), and [seq](#).

**8.142.3.30 add\_series()**

```
ex GiNaC::pseries::add_series (
    const pseries & other) const
```

Add one series object to another, producing a pseries object that represents the sum.

**Parameters**

<i>other</i>	pseries object to add with
--------------	----------------------------

**Returns**

the sum as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [is\\_compatible\\_to\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#).

**8.142.3.31 mul\_const()**

```
ex GiNaC::pseries::mul_const (
    const numeric & other) const
```

Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	constant to multiply with
--------------	---------------------------

**Returns**

the product as a pseries

References [GiNaC::numeric::coeff\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

**8.142.3.32 mul\_series()**

```
ex GiNaC::pseries::mul_series (
    const pseries & other) const
```

Multiply one pseries object to another, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	pseries object to multiply with
--------------	---------------------------------

**Returns**

the product as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [degree\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [is\\_compatible\\_to\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [ldegree\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

**8.142.3.33 power\_const()**

```
ex GiNaC::pseries::power_const (
    const numeric & p,
    int deg) const
```

Compute the p-th power of a series.

**Parameters**

<i>p</i>	power to compute
<i>deg</i>	truncation order of series calculation

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [degree\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [point](#), [GiNaC::pow\(\)](#), [pseries\(\)](#), [GiNaC::numeric::real\(\)](#), [seq](#), [GiNaC::to\\_int\(\)](#), and [var](#).

**8.142.3.34 shift\_exponents()**

```
pseries GiNaC::pseries::shift_exponents (
    int deg) const
```

Return a new pseries object with the powers shifted by deg.

References [point](#), [pseries\(\)](#), [seq](#), and [var](#).



### 8.142.3.35 print\_series()

```
void GiNaC::pseries::print_series (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    const char * pow_sym,
    unsigned level) const [protected]
```

References [GiNaC::ex1](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [point](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::print\\_context::s](#), [seq](#), and [var](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

### 8.142.3.36 do\_print()

```
void GiNaC::pseries::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [print\\_series\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.142.3.37 do\_print\_latex()

```
void GiNaC::pseries::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [print\\_series\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.142.3.38 do\_print\_tree()

```
void GiNaC::pseries::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [point](#), [GiNaC::print\\_context::s](#), [seq](#), and [var](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.142.3.39 do\_print\_python()

```
void GiNaC::pseries::do_print_python (
    const print\_python & c,
    unsigned level) const [protected]
```

References [print\\_series\(\)](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.142.3.40 do\_print\_python\_repr()

```
void GiNaC::pseries::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [point](#), [GiNaC::print\\_context::s](#), [seq](#), and [var](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.142.4 Member Data Documentation

### 8.142.4.1 seq

```
epvector GiNaC::pseries::seq [protected]
```

Vector of {coefficient, power} pairs.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [coeffop\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [exponop\(\)](#), [imag\\_part\(\)](#), [is\\_terminating\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [nops\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

### 8.142.4.2 var

```
ex GiNaC::pseries::var [protected]
```

Series variable (holds a symbol).

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_var\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

### 8.142.4.3 point

`ex` GiNaC::pseries::point [protected]

Expansion point.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_point\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [pseries.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 8.143 GiNaC::psi1\_SERIAL Class Reference

Polylogarithm and multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.143.1 Detailed Description

Polylogarithm and multiple polylogarithm.

Nielsen's generalized polylogarithm. Harmonic polylogarithm. Gamma-function. Beta-function. Psi-function (aka digamma-function).

### 8.143.2 Member Data Documentation

#### 8.143.2.1 serial

```
unsigned GiNaC::psi1_SERIAL::serial [static]
```

**Initial value:**

```
=
function::register_new(function_options("psi", 1).
    eval_func(psi1_eval).
    evalf_func(psi1_evalf).
    derivative_func(psi1_deriv).
    series_func(psi1_series).
    latex_name("\\psi").
    overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 8.144 GiNaC::psi2\_SERIAL Class Reference

Derivatives of Psi-function (aka polygamma-functions).

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.144.1 Detailed Description

Derivatives of Psi-function (aka polygamma-functions).

### 8.144.2 Member Data Documentation

#### 8.144.2.1 serial

```
unsigned GiNaC::psi2_SERIAL::serial [static]
```

**Initial value:**

```
=
function::register_new(function_options("psi", 2).
    eval_func(psi2_eval).
    evalf_func(psi2_evalf).
    derivative_func(psi2_deriv).
    series_func(psi2_series).
    latex_name("\\psi").
    overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

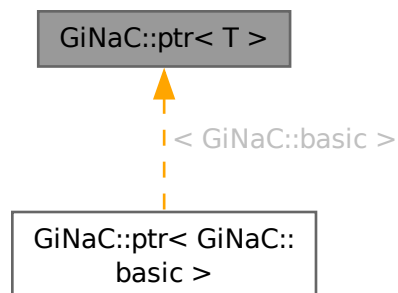
- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 8.145 GiNaC::ptr< T > Class Template Reference

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

```
#include <ptr.h>
```

Inheritance diagram for GiNaC::ptr< T >:



## Public Member Functions

- `ptr (T *t)` noexcept  
*Bind ptr to newly created object, start reference counting.*
- `ptr (T &t)` noexcept  
*Bind ptr to existing reference-counted object.*
- `ptr (const ptr &other)` noexcept
- `~ptr ()`
- `ptr & operator= (const ptr &other)`
- `T & operator* ()` const noexcept
- `T * operator-> ()` const noexcept
- `void makewritable ()`  
*Announce your intention to modify the object bound to this ptr.*
- `void swap (ptr &other)` noexcept  
*Swap the bound object of this ptr with another ptr.*
- `template<class U>`  
`bool operator== (const ptr< U > &rhs)` const noexcept
- `template<class U>`  
`bool operator!= (const ptr< U > &rhs)` const noexcept

## Private Attributes

- `T * p`

## Friends

- `struct std::less< ptr< T > >`
- `T * get_pointer (const ptr &x)` noexcept
- `template<class U>`  
`bool operator== (const ptr &lhs, const U *rhs)` noexcept
- `template<class U>`  
`bool operator!= (const ptr &lhs, const U *rhs)` noexcept
- `template<class U>`  
`bool operator== (const U *lhs, const ptr &rhs)` noexcept
- `template<class U>`  
`bool operator!= (const U *lhs, const ptr &rhs)` noexcept
- `std::ostream & operator<< (std::ostream &os, const ptr< T > &rhs)`

### 8.145.1 Detailed Description

**template<class T>**  
**class GiNaC::ptr< T >**

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

Requirements for T: must support the refcounted interface (usually by being derived from refcounted) T\* T::↵ duplicate() member function (only if makewritable() is used)

## 8.145.2 Constructor & Destructor Documentation

### 8.145.2.1 ptr() [1/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    T * t) [inline], [noexcept]
```

Bind ptr to newly created object, start reference counting.

References [GINAC\\_ASSERT](#), and [p](#).

Referenced by [get\\_pointer](#), [operator!=](#), [operator!==\(](#), [operator!=](#), [operator<<](#), [operator=\(\)](#), [operator==](#), [operator==\(](#), [operator==](#), [ptr\(\)](#), and [swap\(\)](#).

### 8.145.2.2 ptr() [2/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    T & t) [inline], [explicit], [noexcept]
```

Bind ptr to existing reference-counted object.

References [p](#).

### 8.145.2.3 ptr() [3/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
    const ptr< T > & other) [inline], [noexcept]
```

References [p](#), and [ptr\(\)](#).

### 8.145.2.4 ~ptr()

```
template<class T>
GiNaC::ptr< T >::~~ptr () [inline]
```

References [p](#).

## 8.145.3 Member Function Documentation

### 8.145.3.1 operator=()

```
template<class T>
ptr & GiNaC::ptr< T >::operator= (
    const ptr< T > & other) [inline]
```

References [p](#), and [ptr\(\)](#).

**8.145.3.2 operator\*()**

```
template<class T>
T & GiNaC::ptr< T >::operator* () const [inline], [noexcept]
```

References [p](#).

**8.145.3.3 operator->()**

```
template<class T>
T * GiNaC::ptr< T >::operator-> () const [inline], [noexcept]
```

References [p](#).

**8.145.3.4 makewritable()**

```
template<class T>
void GiNaC::ptr< T >::makewritable () [inline]
```

Announce your intention to modify the object bound to this ptr.

This ensures that the object is not shared by any other ptrs.

References [p](#).

**8.145.3.5 swap()**

```
template<class T>
void GiNaC::ptr< T >::swap (
    ptr< T > & other) [inline], [noexcept]
```

Swap the bound object of this ptr with another ptr.

References [p](#), and [ptr\(\)](#).

**8.145.3.6 operator==()**

```
template<class T>
template<class U>
bool GiNaC::ptr< T >::operator== (
    const ptr< U > & rhs) const [inline], [noexcept]
```

References [get\\_pointer](#), [p](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

**8.145.3.7 operator"!="()**

```
template<class T>
template<class U>
bool GiNaC::ptr< T >::operator!= (
    const ptr< U > & rhs) const [inline], [noexcept]
```

References [get\\_pointer](#), [p](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

## 8.145.4 Friends And Related Symbol Documentation

### 8.145.4.1 `std::less< ptr< T > >`

```
template<class T>
friend struct std::less< ptr< T > > [friend]
```

### 8.145.4.2 `get_pointer`

```
template<class T>
T * get_pointer (
    const ptr< T > & x) [friend]
```

References [ptr\(\)](#).

Referenced by [operator!=\(\)](#), and [operator==\(\)](#).

### 8.145.4.3 `operator==` [1/2]

```
template<class T>
template<class U>
bool operator== (
    const ptr< T > & lhs,
    const U * rhs) [friend]
```

References [GiNaC::lhs\(\)](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

### 8.145.4.4 `operator"!="` [1/2]

```
template<class T>
template<class U>
bool operator!= (
    const ptr< T > & lhs,
    const U * rhs) [friend]
```

References [GiNaC::lhs\(\)](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

### 8.145.4.5 `operator==` [2/2]

```
template<class T>
template<class U>
bool operator== (
    const U * lhs,
    const ptr< T > & rhs) [friend]
```

References [GiNaC::lhs\(\)](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).



#### 8.145.4.6 operator"!=" [2/2]

```
template<class T>
template<class U>
bool operator!= (
    const U * lhs,
    const ptr< T > & rhs) [friend]
```

References [GiNaC::lhs\(\)](#), [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

#### 8.145.4.7 operator<<

```
template<class T>
std::ostream & operator<< (
    std::ostream & os,
    const ptr< T > & rhs) [friend]
```

References [ptr\(\)](#), and [GiNaC::rhs\(\)](#).

### 8.145.5 Member Data Documentation

#### 8.145.5.1 p

```
template<class T>
T* GiNaC::ptr< T >::p [private]
```

Referenced by [makewritable\(\)](#), [operator!=\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), [operator=\(\)](#), [operator==\(\)](#), [ptr\(\)](#), [ptr\(\)](#), [ptr\(\)](#), [swap\(\)](#), and [~ptr\(\)](#).

The documentation for this class was generated from the following file:

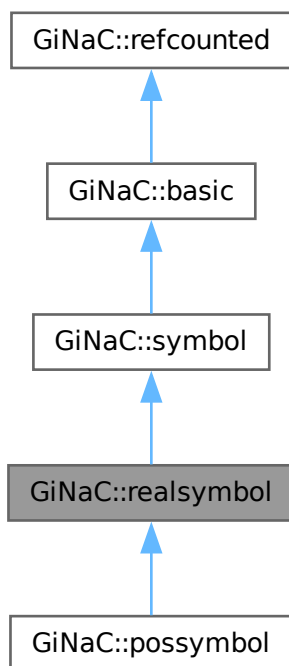
- [ptr.h](#)

## 8.146 GiNaC::realsymbol Class Reference

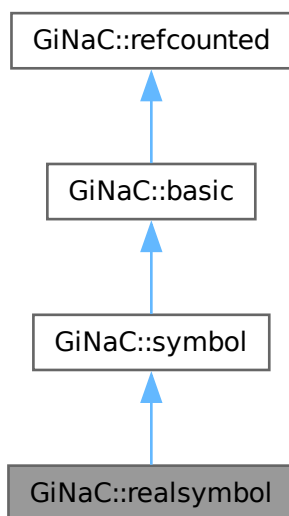
Specialization of symbol to real domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::realsymbol:



Collaboration diagram for GiNaC::realsymbol:



## Public Member Functions

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

## Public Member Functions inherited from [GiNaC::symbol](#)

- [symbol](#) (const std::string &initname)
  - [symbol](#) (const std::string &initname, const std::string &texname)
  - bool [info](#) (unsigned inf) const override
- Information about the object.*
- [ex eval](#) () const override
- Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override
- Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int order, unsigned options=0) const override
- Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override
- Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override
- Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override
- Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override
- Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
  - [ex real\\_part](#) () const override
  - [ex imag\\_part](#) () const override
  - bool [is\\_polynomial](#) (const [ex](#) &var) const override
- Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override
- Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override
- Read (a.k.a.*
- void [set\\_name](#) (const std::string &n)
  - void [set\\_TeX\\_name](#) (const std::string &n)
  - std::string [get\\_name](#) () const
  - std::string [get\\_TeX\\_name](#) () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type ()` const
- virtual `return_type_t return_type_tinfo ()` const
- template<class T>  
void `print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

## Additional Inherited Members

## Protected Member Functions inherited from `GiNaC::symbol`

- `ex derivative (const symbol &s)` const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- bool `is_equal_same_type (const basic &other)` const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash ()` const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print (const print_context &c, unsigned level)` const
- void `do_print_latex (const print_latex &c, unsigned level)` const
- void `do_print_tree (const print_tree &c, unsigned level)` const
- void `do_print_python_repr (const print_python_repr &c, unsigned level)` const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes inherited from [GiNaC::symbol](#)

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.146.1 Detailed Description

Specialization of symbol to real domain.

### 8.146.2 Constructor & Destructor Documentation

#### 8.146.2.1 [realsymbol\(\)](#) [1/3]

`GiNaC::realsymbol::realsymbol ()`

References [realsymbol\(\)](#), and [GiNaC::symbol::symbol\(\)](#).

Referenced by [duplicate\(\)](#), [GiNaC::possymbol::possymbol\(\)](#), [GiNaC::possymbol::possymbol\(\)](#), [GiNaC::possymbol::possymbol\(\)](#), and [realsymbol\(\)](#).

### 8.146.2.2 realsymbol() [2/3]

```
GiNaC::realsymbol::realsymbol (  
    const std::string & initname) [explicit]
```

References [GiNaC::symbol::symbol\(\)](#).

### 8.146.2.3 realsymbol() [3/3]

```
GiNaC::realsymbol::realsymbol (  
    const std::string & initname,  
    const std::string & texname)
```

References [GiNaC::symbol::symbol\(\)](#).

## 8.146.3 Member Function Documentation

### 8.146.3.1 get\_domain()

```
unsigned GiNaC::realsymbol::get_domain () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

References [GiNaC::domain::real](#).

### 8.146.3.2 conjugate()

```
ex GiNaC::realsymbol::conjugate () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 8.146.3.3 real\_part()

```
ex GiNaC::realsymbol::real_part () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 8.146.3.4 imag\_part()

```
ex GiNaC::realsymbol::imag_part () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 8.146.3.5 `duplicate()`

```
realsymbol * GiNaC::realsymbol::duplicate () const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::dynallocated](#), [realsymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [symbol.cpp](#)

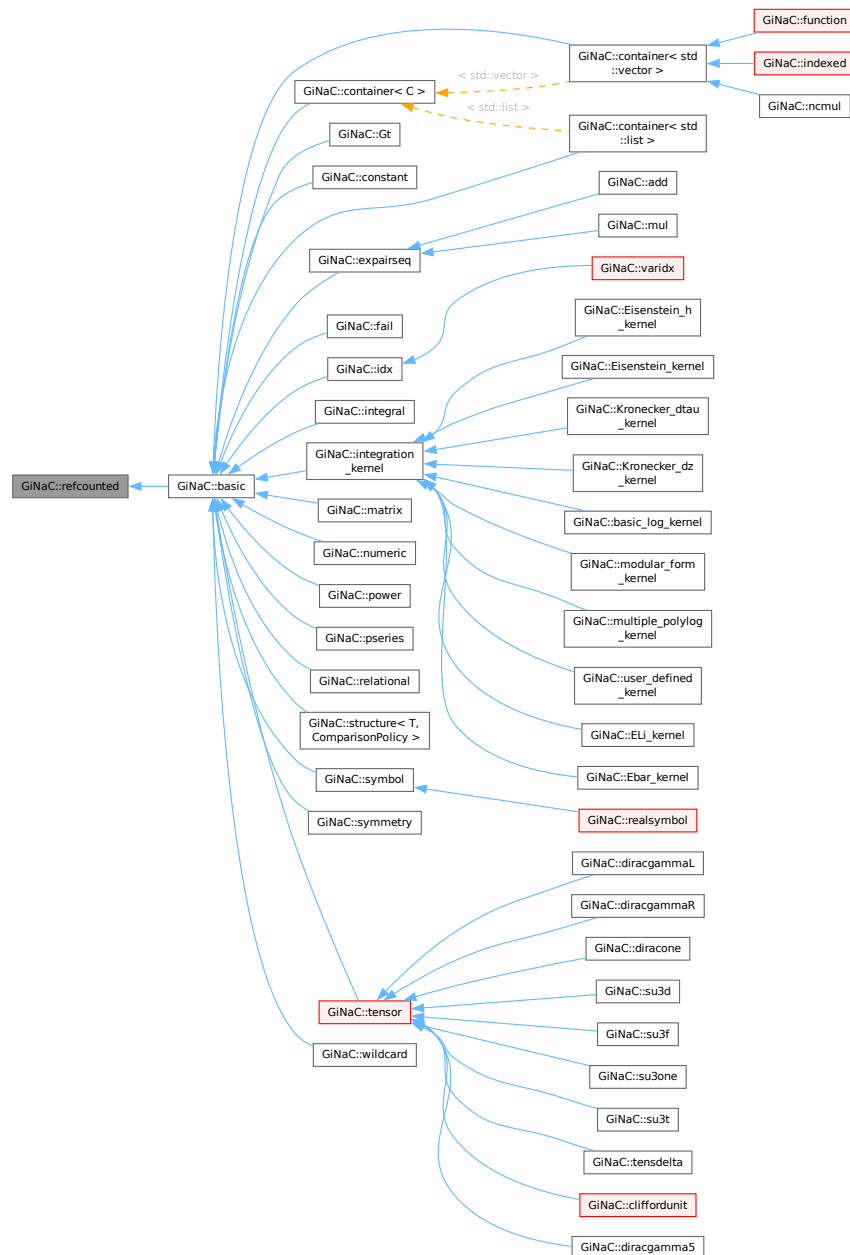
## 8.147 `GiNaC::refcounted` Class Reference

Base class for reference-counted objects.

```
#include <ptr.h>
```



Inheritance diagram for GiNaC::refcounted:



## Public Member Functions

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Private Attributes

- unsigned int [refcount](#)  
reference counter

### 8.147.1 Detailed Description

Base class for reference-counted objects.

### 8.147.2 Constructor & Destructor Documentation

#### 8.147.2.1 `refcounted()`

```
GiNaC::refcounted::refcounted () [inline], [noexcept]
```

References [refcount](#).

### 8.147.3 Member Function Documentation

#### 8.147.3.1 `add_reference()`

```
unsigned int GiNaC::refcounted::add_reference () [inline], [noexcept]
```

References [refcount](#).

#### 8.147.3.2 `remove_reference()`

```
unsigned int GiNaC::refcounted::remove_reference () [inline], [noexcept]
```

References [refcount](#).

#### 8.147.3.3 `get_refcount()`

```
unsigned int GiNaC::refcounted::get_refcount () const [inline], [noexcept]
```

References [refcount](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

#### 8.147.3.4 `set_refcount()`

```
void GiNaC::refcounted::set_refcount (
    unsigned int r) [inline], [noexcept]
```

References [refcount](#).

Referenced by [GiNaC::basic::operator=\(\)](#).

## 8.147.4 Member Data Documentation

### 8.147.4.1 refcount

```
unsigned int GiNaC::refcounted::refcount [private]
```

reference counter

Referenced by [add\\_reference\(\)](#), [get\\_refcount\(\)](#), [refcounted\(\)](#), [remove\\_reference\(\)](#), and [set\\_refcount\(\)](#).

The documentation for this class was generated from the following file:

- [ptr.h](#)

## 8.148 GiNaC::registered\_class\_options Class Reference

This class stores information about a registered [GiNaC](#) class.

```
#include <registrar.h>
```

### Public Member Functions

- [registered\\_class\\_options](#) (const char \*n, const char \*p, const std::type\_info &ti)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- std::type\_info const \* [get\\_id](#) () const
- const std::vector< [print\\_func](#) > & [get\\_print\\_dispatch\\_table](#) () const
- template<class Ctx, class T, class C>  
[registered\\_class\\_options](#) & [print\\_func](#) (void f(const T &, const C &c, unsigned))
- template<class Ctx, class T, class C>  
[registered\\_class\\_options](#) & [print\\_func](#) (void(T::\*f)(const C &, unsigned))
- template<class Ctx>  
[registered\\_class\\_options](#) & [print\\_func](#) (const [print\\_func](#) &f)
- void [set\\_print\\_func](#) (unsigned id, const [print\\_func](#) &f)

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- std::type\_info const \* [tinfo\\_key](#)  
*Type information key.*
- std::vector< [print\\_func](#) > [print\\_dispatch\\_table](#)  
*Method table for print() dispatch.*

### 8.148.1 Detailed Description

This class stores information about a registered [GiNaC](#) class.

## 8.148.2 Constructor & Destructor Documentation

### 8.148.2.1 registered\_class\_options()

```
GiNaC::registered_class_options::registered_class_options (  
    const char * n,  
    const char * p,  
    const std::type_info & ti) [inline]
```

References [name](#), [parent\\_name](#), and [tinfo\\_key](#).

Referenced by [print\\_func\(\)](#), [print\\_func\(\)](#), and [print\\_func\(\)](#).

## 8.148.3 Member Function Documentation

### 8.148.3.1 get\_name()

```
const char * GiNaC::registered_class_options::get_name () const [inline]
```

References [name](#).

### 8.148.3.2 get\_parent\_name()

```
const char * GiNaC::registered_class_options::get_parent_name () const [inline]
```

References [parent\\_name](#).

### 8.148.3.3 get\_id()

```
std::type_info const * GiNaC::registered_class_options::get_id () const [inline]
```

References [tinfo\\_key](#).

### 8.148.3.4 get\_print\_dispatch\_table()

```
const std::vector< print\_func\_t > & GiNaC::registered_class_options::get_print_dispatch_table  
( ) const [inline]
```

References [print\\_dispatch\\_table](#).

### 8.148.3.5 print\_func() [1/3]

```
template<class Ctx, class T, class C>  
registered\_class\_options & GiNaC::registered_class_options::print_func (  
    void fconst T &, const C &c, unsigned) [inline]
```

References [registered\\_class\\_options\(\)](#), and [set\\_print\\_func\(\)](#).

**8.148.3.6 print\_func()** [2/3]

```
template<class Ctx, class T, class C>
registered\_class\_options & GiNaC::registered_class_options::print_func (
    void(T::* f) (const C &, unsigned)) [inline]
```

References [registered\\_class\\_options\(\)](#), and [set\\_print\\_func\(\)](#).

**8.148.3.7 print\_func()** [3/3]

```
template<class Ctx>
registered\_class\_options & GiNaC::registered_class_options::print_func (
    const print\_functor & f) [inline]
```

References [registered\\_class\\_options\(\)](#), and [set\\_print\\_func\(\)](#).

**8.148.3.8 set\_print\_func()**

```
void GiNaC::registered_class_options::set_print_func (
    unsigned id,
    const print\_functor & f) [inline]
```

References [print\\_dispatch\\_table](#).

Referenced by [print\\_func\(\)](#), [print\\_func\(\)](#), and [print\\_func\(\)](#).

**8.148.4 Member Data Documentation****8.148.4.1 name**

```
const char* GiNaC::registered_class_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#), and [registered\\_class\\_options\(\)](#).

**8.148.4.2 parent\_name**

```
const char* GiNaC::registered_class_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#), and [registered\\_class\\_options\(\)](#).

**8.148.4.3 tinfo\_key**

```
std::type_info const* GiNaC::registered_class_options::tinfo_key [private]
```

Type information key.

Referenced by [get\\_id\(\)](#), and [registered\\_class\\_options\(\)](#).

#### 8.148.4.4 print\_dispatch\_table

```
std::vector<print_functor> GiNaC::registered_class_options::print_dispatch_table [private]
```

Method table for print() dispatch.

Referenced by [get\\_print\\_dispatch\\_table\(\)](#), and [set\\_print\\_func\(\)](#).

The documentation for this class was generated from the following file:

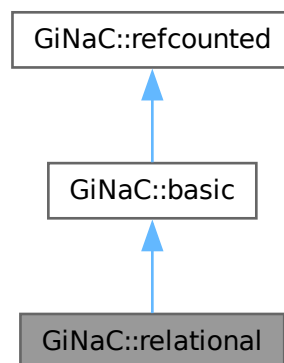
- [registrar.h](#)

### 8.149 GiNaC::relational Class Reference

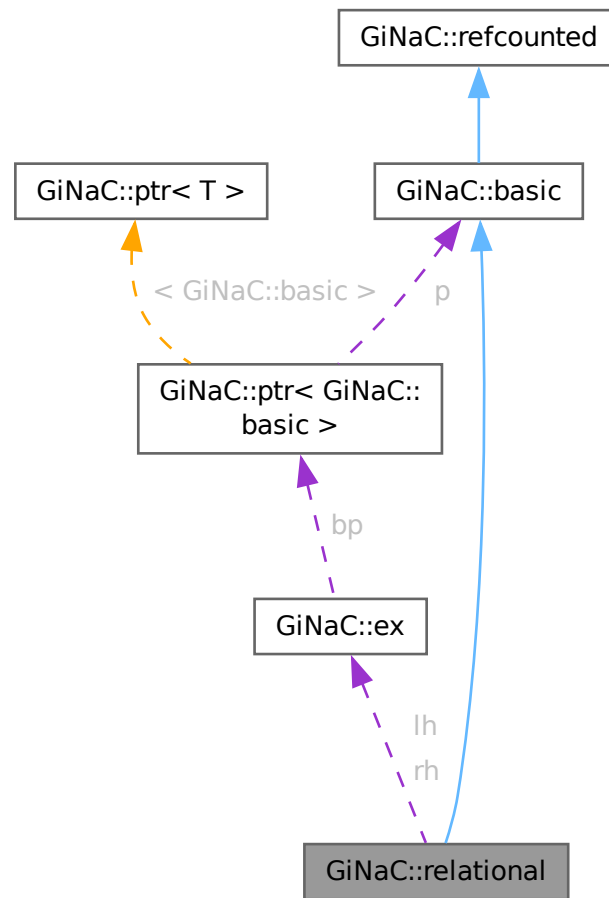
This class holds a relation consisting of two expressions and a logical relation between them.

```
#include <relational.h>
```

Inheritance diagram for GiNaC::relational:



Collaboration diagram for GiNaC::relational:



## Classes

- struct [safe\\_bool\\_helper](#)

## Public Types

- enum [operators](#) {  
[equal](#) , [not\\_equal](#) , [less](#) , [less\\_or\\_equal](#) ,  
[greater](#) , [greater\\_or\\_equal](#) }

## Public Member Functions

- [relational](#) (const [ex](#) &[lhs](#), const [ex](#) &[rhs](#), [operators](#) oper=[equal](#))
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthesizing output).*
- bool [info](#) (unsigned inf) const override

- Information about the object.*
- `size_t nops ()` const override
- Number of operands/members.*
- `ex op (size_t i)` const override
- Return operand/member at position i.*
- `ex map (map_function &f)` const override
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex subs (const exmap &m, unsigned options=0)` const override
- Substitute a set of objects by arbitrary expressions.*
- `void archive (archive_node &n)` const override
- Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &syms)` override
- Read (a.k.a.*
- `ex canonical ()` const
- Returns an equivalent relational with zero right-hand side.*
- `ex lhs ()` const
- `ex rhs ()` const
- `operator safe_bool ()` const
- Cast the relational into a Boolean, mainly for evaluation within an if-statement.*
- `safe_bool operator! ()` const

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`
- basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`
- basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate ()` const
- Create a clone of this object on the heap.*
- `virtual ex eval ()` const
- Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf ()` const
- Evaluate object numerically.*
- `virtual ex evalm ()` const
- Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ ()` const
- Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i)` const
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0)` const
- Output to stream.*
- `virtual void dbgprint ()` const
- Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree ()` const
- Little wrapper around printtree to be called within a debugger.*
- `virtual ex operator[] (const ex &index)` const
- `virtual ex operator[] (size_t i)` const
- `virtual ex & let_op (size_t i)`
- Return modifiable operand/member at position i.*



- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const relational &r, int order, unsigned options=0) const  
*Default implementation of ex::series().*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of ex::normal().*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of ex::to\_rational().*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation ex::max\_coefficient().*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like print(), but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for subs().*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative ex::diff(s, n).*

- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes**

- [ex lh](#)
- [ex rh](#)
- [operators o](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

**Private Types**

- typedef void(safe\_bool\_helper::\* [safe\\_bool](#)) ()

**Private Member Functions**

- [safe\\_bool make\\_safe\\_bool](#) (bool) const

**8.149.1 Detailed Description**

This class holds a relation consisting of two expressions and a logical relation between them.

**8.149.2 Member Typedef Documentation****8.149.2.1 [safe\\_bool](#)**

```
typedef void(safe_bool_helper::* GiNaC::relational::safe\_bool) () [private]
```

**8.149.3 Member Enumeration Documentation****8.149.3.1 [operators](#)**

```
enum GiNaC::relational::operators
```

**Enumerator**

<a href="#">equal</a>	
<a href="#">not_equal</a>	
<a href="#">less</a>	
<a href="#">less_or_equal</a>	

greater	
greater_or_equal	

## 8.149.4 Constructor & Destructor Documentation

### 8.149.4.1 relational()

```
GiNaC::relational::relational (
    const ex & lhs,
    const ex & rhs,
    operators oper = equal)
```

References [GiNaC::basic::ex](#), [lh](#), [lhs\(\)](#), [o](#), [rh](#), and [rhs\(\)](#).

Referenced by [canonical\(\)](#), [match\\_same\\_type\(\)](#), and [subs\(\)](#).

## 8.149.5 Member Function Documentation

### 8.149.5.1 precedence()

```
unsigned GiNaC::relational::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\(\)](#).

### 8.149.5.2 info()

```
bool GiNaC::relational::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [equal](#), [greater](#), [greater\\_or\\_equal](#), [less](#), [less\\_or\\_equal](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::relation](#), [GiNaC::info\\_flags::relation\\_equal](#), [GiNaC::info\\_flags::relation\\_greater](#), [GiNaC::info\\_flags::relation\\_greater\\_or\\_equal](#), [GiNaC::info\\_flags::relation\\_less](#), [GiNaC::info\\_flags::relation\\_less\\_or\\_equal](#), and [GiNaC::info\\_flags::relation\\_not\\_equal](#).

Referenced by [operator safe\\_bool\(\)](#).

### 8.149.5.3 nops()

```
size_t GiNaC::relational::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.149.5.4 op()

```
ex GiNaC::relational::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [lh](#), and [rh](#).

### 8.149.5.5 map()

```
ex GiNaC::relational::map (
    map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::dynallocate\(\)](#), [GiNaC::basic::ex](#), [lh](#), [o](#), and [rh](#).

### 8.149.5.6 subs()

```
ex GiNaC::relational::subs (
    const exmap & m,
    unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::ex](#), [lh](#), [o](#), [relational\(\)](#), [rh](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 8.149.5.7 archive()

```
void GiNaC::relational::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [lh](#), [o](#), and [rh](#).

### 8.149.5.8 read\_archive()

```
void GiNaC::relational::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [lh](#), [o](#), and [rh](#).

### 8.149.5.9 canonical()

```
ex GiNaC::relational::canonical () const
```

Returns an equivalent relational with zero right-hand side.

References [GiNaC::\\_ex0](#), [GiNaC::basic::ex](#), [lh](#), [o](#), [relational\(\)](#), and [rh](#).

### 8.149.5.10 eval\_ncmul()

```
ex GiNaC::relational::eval_ncmul (
    const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [lh](#).

### 8.149.5.11 match\_same\_type()

```
bool GiNaC::relational::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [o](#), and [relational\(\)](#).

**8.149.5.12 return\_type()**

```
unsigned GiNaC::relational::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), and [rh](#).

**8.149.5.13 return\_type\_tinfo()**

```
return_type_t GiNaC::relational::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), and [rh](#).

**8.149.5.14 calchash()**

```
unsigned GiNaC::relational::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [equal](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::status\\_flags::hash\\_calcul](#), [GiNaC::basic::hashvalue](#), [less](#), [less\\_or\\_equal](#), [lh](#), [not\\_equal](#), [o](#), [rh](#), [GiNaC::rotate\\_left\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**8.149.5.15 do\_print()**

```
void GiNaC::relational::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [lh](#), [o](#), [precedence\(\)](#), [GiNaC::print\\_operator\(\)](#), [rh](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.149.5.16 do\_print\_python\_repr()**

```
void GiNaC::relational::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [lh](#), [o](#), [GiNaC::print\\_operator\(\)](#), [rh](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

**8.149.5.17 lhs()**

```
ex GiNaC::relational::lhs () const [inline]
```

References [GiNaC::basic::ex](#), and [lh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_series\(\)](#), [relational\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::tan\\_series\(\)](#), and [GiNaC::tanh\\_series\(\)](#).

**8.149.5.18 rhs()**

```
ex GiNaC::relational::rhs () const [inline]
```

References [GiNaC::basic::ex](#), and [rh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [relational\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::pseries::series\(\)](#), and [GiNaC::symbol::series\(\)](#).

**8.149.5.19 make\_safe\_bool()**

```
relational::safe_bool GiNaC::relational::make_safe_bool (
    bool cond) const [private]
```

References [GiNaC::relational::safe\\_bool\\_helper::nonnull\(\)](#).

Referenced by [operator safe\\_bool\(\)](#), and [operator!\(\)](#).

**8.149.5.20 operator safe\_bool()**

```
GiNaC::relational::operator relational::safe_bool () const
```

Cast the relational into a Boolean, mainly for evaluation within an if-statement.

Note that  $(a < b) == \text{false}$  does not imply  $(a \geq b) == \text{true}$  in the general symbolic case. A false result means the comparison is either false or undecidable (except of course for  $!=$ , where true means either unequal or undecidable).

References [GiNaC::\\_num0\\_p](#), [equal](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [less](#), [less\\_or\\_equal](#), [lh](#), [make\\_safe\\_bool\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::positive](#), and [rh](#).

**8.149.5.21 operator"!()**

```
relational::safe_bool GiNaC::relational::operator! () const [inline]
```

References [make\\_safe\\_bool\(\)](#).



## 8.149.6 Member Data Documentation

### 8.149.6.1 lh

`ex` GiNaC::relational::lh [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\\_ncmul\(\)](#), [lhs\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [relational\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [subs\(\)](#).

### 8.149.6.2 rh

`ex` GiNaC::relational::rh [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [relational\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), and [subs\(\)](#).

### 8.149.6.3 o

`operators` GiNaC::relational::o [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [info\(\)](#), [map\(\)](#), [match\\_same\\_type\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [relational\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [relational.h](#)
- [relational.cpp](#)

## 8.150 GiNaC::remember\_strategies Class Reference

Strategies how to clean up the function remember cache.

```
#include <flags.h>
```

### Public Types

- enum { [delete\\_never](#) , [delete\\_lru](#) , [delete\\_lfu](#) , [delete\\_cyclic](#) }

### 8.150.1 Detailed Description

Strategies how to clean up the function remember cache.

See also

[remember\\_table](#)

## 8.150.2 Member Enumeration Documentation

### 8.150.2.1 anonymous enum

anonymous enum

#### Enumerator

<code>delete_never</code>	Let table grow indefinitely.
---------------------------	------------------------------

delete_lru	Least recently used.
delete_lfu	Least frequently used.
delete_cyclic	First (oldest) one in list.

The documentation for this class was generated from the following file:

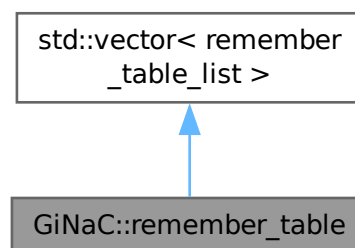
- [flags.h](#)

## 8.151 GiNaC::remember\_table Class Reference

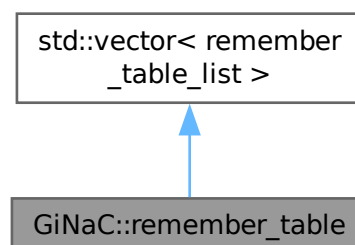
The remember table is organized like an n-fold associative cache in a microprocessor.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table:



Collaboration diagram for GiNaC::remember\_table:



## Public Member Functions

- [remember\\_table](#) ()
- [remember\\_table](#) (unsigned s, unsigned as, unsigned strat)
- bool [lookup\\_entry](#) (function const &f, [ex](#) &result) const
- void [add\\_entry](#) (function const &f, [ex](#) const &result)
- void [clear\\_all\\_entries](#) ()
- void [show\\_statistics](#) (std::ostream &os, unsigned level) const

## Static Public Member Functions

- static std::vector< [remember\\_table](#) > & [remember\\_tables](#) ()

## Protected Member Functions

- void [init\\_table](#) ()

## Protected Attributes

- unsigned [table\\_size](#)
- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 8.151.1 Detailed Description

The remember table is organized like an n-fold associative cache in a microprocessor.

The table has a width of 's' (which is rounded to [table\\_size](#), some power of 2 near 's', internally) and a depth of 'as' (unless you choose that entries are never discarded). The place where an entry is stored depends on the hashvalue of the parameters of the function (this corresponds to the address of byte to be cached). The '[log\\_2\(table\\_size\)](#)' least significant bits of this hashvalue give the slot in which the entry will be stored or looked up. Each slot can take up to 'as' entries. If a slot is full, an older entry is removed by one of the following strategies:

- oldest entry (the first one in the list)
- least recently used (the one with the lowest 'last\_access')
- least frequently used (the one with the lowest 'successful\_hits') or all entries are kept which means that the table grows indefinitely.

### 8.151.2 Constructor & Destructor Documentation

#### 8.151.2.1 [remember\\_table](#)() [1/2]

```
GiNaC::remember_table::remember_table ()
```

References [GiNaC::remember\\_strategies::delete\\_never](#), [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

### 8.151.2.2 `remember_table()` [2/2]

```
GiNaC::remember_table::remember_table (
    unsigned s,
    unsigned as,
    unsigned strat)
```

References [init\\_table\(\)](#), [GiNaC::log2\(\)](#), [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

## 8.151.3 Member Function Documentation

### 8.151.3.1 `lookup_entry()`

```
bool GiNaC::remember_table::lookup_entry (
    function const & f,
    ex & result) const
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 8.151.3.2 `add_entry()`

```
void GiNaC::remember_table::add_entry (
    function const & f,
    ex const & result)
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 8.151.3.3 `clear_all_entries()`

```
void GiNaC::remember_table::clear_all_entries ()
```

References [init\\_table\(\)](#).

### 8.151.3.4 `show_statistics()`

```
void GiNaC::remember_table::show_statistics (
    std::ostream & os,
    unsigned level) const
```

### 8.151.3.5 `remember_tables()`

```
std::vector< remember_table > & GiNaC::remember_table::remember_tables () [static]
```

Referenced by [GiNaC::function::lookup\\_remember\\_table\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [GiNaC::function::store\\_remember\\_t](#)

### 8.151.3.6 init\_table()

```
void GiNaC::remember_table::init_table () [protected]
```

References [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

Referenced by [clear\\_all\\_entries\(\)](#), and [remember\\_table\(\)](#).

## 8.151.4 Member Data Documentation

### 8.151.4.1 table\_size

```
unsigned GiNaC::remember_table::table_size [protected]
```

Referenced by [add\\_entry\(\)](#), [init\\_table\(\)](#), [lookup\\_entry\(\)](#), [remember\\_table\(\)](#), and [remember\\_table\(\)](#).

### 8.151.4.2 max\_assoc\_size

```
unsigned GiNaC::remember_table::max_assoc_size [protected]
```

Referenced by [init\\_table\(\)](#), [remember\\_table\(\)](#), and [remember\\_table\(\)](#).

### 8.151.4.3 remember\_strategy

```
unsigned GiNaC::remember_table::remember_strategy [protected]
```

Referenced by [init\\_table\(\)](#), [remember\\_table\(\)](#), and [remember\\_table\(\)](#).

The documentation for this class was generated from the following files:

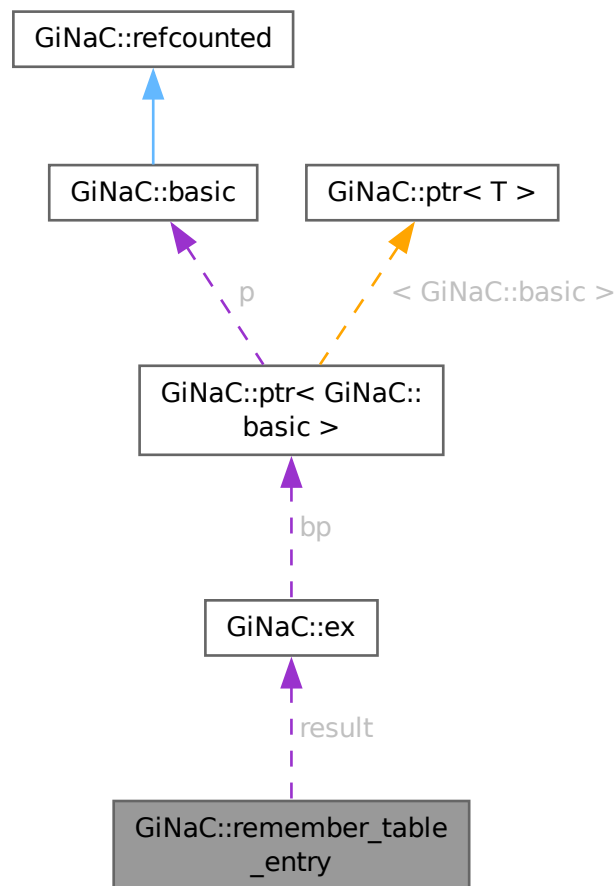
- [remember.h](#)
- [remember.cpp](#)

## 8.152 GiNaC::remember\_table\_entry Class Reference

A single entry in the remember table of a function.

```
#include <remember.h>
```

Collaboration diagram for GiNaC::remember\_table\_entry:



### Public Member Functions

- `remember_table_entry` (function const &f, ex const &r)
- bool `is_equal` (function const &f) const
- ex `get_result` () const
- unsigned long `get_last_access` () const
- unsigned long `get_successful_hits` () const

### Protected Attributes

- unsigned `hashvalue`
- `exvector` `seq`
- ex `result`
- unsigned long `last_access`
- unsigned `successful_hits`

## Static Protected Attributes

- static unsigned long [access\\_counter](#) = 0

### 8.152.1 Detailed Description

A single entry in the remember table of a function.

Needs to be a friend of class function to access 'seq'. 'last\_access' and 'successful\_hits' are updated at each successful 'is\_equal'.

### 8.152.2 Constructor & Destructor Documentation

#### 8.152.2.1 remember\_table\_entry()

```
GiNaC::remember_table_entry::remember_table_entry (  
    function const & f,  
    ex const & r)
```

References [access\\_counter](#), [hashvalue](#), [last\\_access](#), [result](#), [seq](#), and [successful\\_hits](#).

### 8.152.3 Member Function Documentation

#### 8.152.3.1 is\_equal()

```
bool GiNaC::remember_table_entry::is_equal (  
    function const & f) const
```

References [access\\_counter](#), [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), [hashvalue](#), [is\\_equal\(\)](#), [last\\_access](#), [GiNaC::container\\_storage< C >::seq](#), [seq](#), and [successful\\_hits](#).

Referenced by [is\\_equal\(\)](#).

#### 8.152.3.2 get\_result()

```
ex GiNaC::remember_table_entry::get_result () const [inline]
```

References [result](#).

#### 8.152.3.3 get\_last\_access()

```
unsigned long GiNaC::remember_table_entry::get_last_access () const [inline]
```

References [last\\_access](#).

#### 8.152.3.4 `get_successful_hits()`

```
unsigned long GiNaC::remember_table_entry::get_successful_hits () const [inline]
```

References [successful\\_hits](#).

### 8.152.4 Member Data Documentation

#### 8.152.4.1 `hashvalue`

```
unsigned GiNaC::remember_table_entry::hashvalue [protected]
```

Referenced by [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 8.152.4.2 `seq`

```
exvector GiNaC::remember_table_entry::seq [protected]
```

Referenced by [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 8.152.4.3 `result`

```
ex GiNaC::remember_table_entry::result [protected]
```

Referenced by [get\\_result\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 8.152.4.4 `last_access`

```
unsigned long GiNaC::remember_table_entry::last_access [mutable], [protected]
```

Referenced by [get\\_last\\_access\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 8.152.4.5 `successful_hits`

```
unsigned GiNaC::remember_table_entry::successful_hits [mutable], [protected]
```

Referenced by [get\\_successful\\_hits\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 8.152.4.6 `access_counter`

```
unsigned long GiNaC::remember_table_entry::access_counter = 0 [static], [protected]
```

Referenced by [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

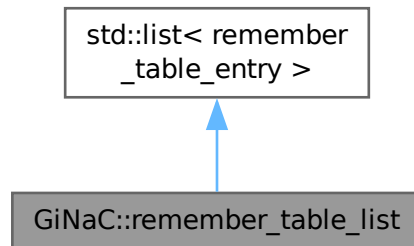


## 8.153 GiNaC::remember\_table\_list Class Reference

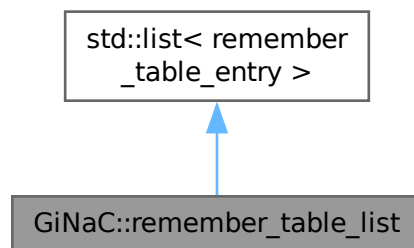
A list of entries in the remember table having some least significant bits of the hashvalue in common.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table\_list:



Collaboration diagram for GiNaC::remember\_table\_list:



### Public Member Functions

- [remember\\_table\\_list](#) (unsigned as, unsigned strat)
- void [add\\_entry](#) (function const &f, ex const &result)
- bool [lookup\\_entry](#) (function const &f, ex &result) const

### Protected Attributes

- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 8.153.1 Detailed Description

A list of entries in the remember table having some least significant bits of the hashvalue in common.

### 8.153.2 Constructor & Destructor Documentation

#### 8.153.2.1 `remember_table_list()`

```
GiNaC::remember_table_list::remember_table_list (  
    unsigned as,  
    unsigned strat)
```

References [max\\_assoc\\_size](#), and [remember\\_strategy](#).

### 8.153.3 Member Function Documentation

#### 8.153.3.1 `add_entry()`

```
void GiNaC::remember_table_list::add_entry (  
    function const & f,  
    ex const & result)
```

References [GiNaC::remember\\_strategies::delete\\_cyclic](#), [GiNaC::remember\\_strategies::delete\\_ifu](#), [GiNaC::remember\\_strategies::delete\\_never](#), [GINAC\\_ASSERT](#), [max\\_assoc\\_size](#), and [remember\\_strategy](#).

#### 8.153.3.2 `lookup_entry()`

```
bool GiNaC::remember_table_list::lookup_entry (  
    function const & f,  
    ex & result) const
```

### 8.153.4 Member Data Documentation

#### 8.153.4.1 `max_assoc_size`

```
unsigned GiNaC::remember_table_list::max_assoc_size [protected]
```

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

#### 8.153.4.2 `remember_strategy`

```
unsigned GiNaC::remember_table_list::remember_strategy [protected]
```

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

## 8.154 GiNaC::return\_type\_t Struct Reference

To distinguish between different kinds of non-commutative objects.

```
#include <registrar.h>
```

### Public Member Functions

- bool [operator<](#) (const [return\\_type\\_t](#) &other) const  
*Strict weak ordering (so one can put [return\\_type\\_t](#)'s into a STL container).*
- bool [operator==](#) (const [return\\_type\\_t](#) &other) const
- bool [operator!=](#) (const [return\\_type\\_t](#) &other) const

### Public Attributes

- std::type\_info const \* [tinfo](#)  
*to distinguish between non-commutative objects of different type.*
- unsigned [rl](#)  
*to distinguish between non-commutative objects of the same type.*

### 8.154.1 Detailed Description

To distinguish between different kinds of non-commutative objects.

### 8.154.2 Member Function Documentation

#### 8.154.2.1 [operator<\(\)](#)

```
bool GiNaC::return_type_t::operator< (  
    const return\_type\_t & other) const [inline]
```

Strict weak ordering (so one can put [return\\_type\\_t](#)'s into a STL container).

References [rl](#), and [tinfo](#).

#### 8.154.2.2 [operator==\(\)](#)

```
bool GiNaC::return_type_t::operator== (  
    const return\_type\_t & other) const [inline]
```

References [rl](#), and [tinfo](#).

Referenced by [operator!=\(\)](#).

### 8.154.2.3 `operator"!=()`

```
bool GiNaC::return_type_t::operator!= (
    const return\_type\_t & other) const [inline]
```

References [operator==\(\(\)\)](#).

## 8.154.3 Member Data Documentation

### 8.154.3.1 `tinfo`

```
std::type_info const* GiNaC::return_type_t::tinfo
```

to distinguish between non-commutative objects of different type.

Referenced by [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::is\\_color\\_tinfo\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(\(\)\)](#), [GiNaC::basic::return\\_type\\_tinfo\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::return\\_type\\_tinfo\(\)](#).

### 8.154.3.2 `rl`

```
unsigned GiNaC::return_type_t::rl
```

to distinguish between non-commutative objects of the same type.

Think of gamma matrices with different representation labels.

Referenced by [GiNaC::get\\_representation\\_label\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(\(\)\)](#), [GiNaC::basic::return\\_type\\_tinfo\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::return\\_type\\_tinfo\(\)](#).

The documentation for this struct was generated from the following file:

- [registrar.h](#)

## 8.155 `GiNaC::return_types` Class Reference

```
#include <flags.h>
```

### Public Types

- enum { [commutative](#) , [noncommutative](#) , [noncommutative\\_composite](#) }

## 8.155.1 Member Enumeration Documentation

### 8.155.1.1 `anonymous enum`

```
anonymous enum
```

#### Enumerator

<code>commutative</code>	
--------------------------	--

noncommutative	
noncommutative_composite	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.156 GiNaC::relational::safe\_bool\_helper Struct Reference

### Public Member Functions

- void [nonnull](#) ()

### 8.156.1 Member Function Documentation

#### 8.156.1.1 nonnull()

```
void GiNaC::relational::safe_bool_helper::nonnull () [inline]
```

Referenced by [GiNaC::relational::make\\_safe\\_bool\(\)](#).

The documentation for this struct was generated from the following file:

- [relational.h](#)

## 8.157 GiNaC::scalar\_products Class Reference

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

```
#include <indexed.h>
```

### Public Member Functions

- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &sp)  
*Register scalar product pair and its value.*
- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim, const [ex](#) &sp)  
*Register scalar product pair and its value for a specific space dimension.*
- void [add\\_vectors](#) (const [lst](#) &l, const [ex](#) &dim=[wild](#)())  
*Register list of vectors.*
- void [clear](#) ()  
*Clear all registered scalar products.*
- bool [is\\_defined](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Check whether scalar product pair is defined.*
- [ex](#) [evaluate](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Return value of defined scalar product pair.*
- void [debugprint](#) () const

## Protected Attributes

- [smap](#) [spm](#)

### 8.157.1 Detailed Description

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

See also

[simplify\\_indexed](#)

### 8.157.2 Member Function Documentation

#### 8.157.2.1 [add\(\)](#) [1/2]

```
void GiNaC::scalar_products::add (  
    const ex & v1,  
    const ex & v2,  
    const ex & sp)
```

Register scalar product pair and its value.

References [spm](#).

Referenced by [add\\_vectors\(\)](#).

#### 8.157.2.2 [add\(\)](#) [2/2]

```
void GiNaC::scalar_products::add (  
    const ex & v1,  
    const ex & v2,  
    const ex & dim,  
    const ex & sp)
```

Register scalar product pair and its value for a specific space dimension.

References [spm](#).

#### 8.157.2.3 [add\\_vectors\(\)](#)

```
void GiNaC::scalar_products::add_vectors (  
    const lst & l,  
    const ex & dim = wild\(\))
```

Register list of vectors.

This adds all possible pairs of products  $a.i * b.i$  with the value  $a*b$  (note that this is not a scalar vector product but an ordinary product of scalars).

References [add\(\)](#).

#### 8.157.2.4 clear()

```
void GiNaC::scalar_products::clear ()
```

Clear all registered scalar products.

References [spm](#).

#### 8.157.2.5 is\_defined()

```
bool GiNaC::scalar_products::is_defined (
    const ex & v1,
    const ex & v2,
    const ex & dim) const
```

Check whether scalar product pair is defined.

References [spm](#).

Referenced by [GiNaC::indexed::simplify\\_indexed\\_product](#).

#### 8.157.2.6 evaluate()

```
ex GiNaC::scalar_products::evaluate (
    const ex & v1,
    const ex & v2,
    const ex & dim) const
```

Return value of defined scalar product pair.

References [spm](#).

Referenced by [GiNaC::indexed::simplify\\_indexed\\_product](#).

#### 8.157.2.7 debugprint()

```
void GiNaC::scalar_products::debugprint () const
```

References [GiNaC::spmapkey::debugprint\(\)](#), and [spm](#).

### 8.157.3 Member Data Documentation

#### 8.157.3.1 spm

```
spmap GiNaC::scalar_products::spm [protected]
```

Referenced by [add\(\)](#), [add\(\)](#), [clear\(\)](#), [debugprint\(\)](#), [evaluate\(\)](#), and [is\\_defined\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 8.158 GiNaC::series\_options Class Reference

Flags to control series expansion.

```
#include <flags.h>
```

### Public Types

- enum { [suppress\\_branchcut](#) = 0x0001 }

### 8.158.1 Detailed Description

Flags to control series expansion.

### 8.158.2 Member Enumeration Documentation

#### 8.158.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

suppress_branchcut	Suppress branch cuts in series expansion. Branch cuts manifest themselves as step functions, if this option is not passed. If it is passed and expansion at a point on a cut is performed, then the analytic continuation of the function is expanded.
--------------------	--

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.159 GiNaC::solve\_algo Class Reference

Switch to control algorithm for linear system solving.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#) , [gauss](#) , [divfree](#) , [bareiss](#) ,  
[markowitz](#) }

### 8.159.1 Detailed Description

Switch to control algorithm for linear system solving.

### 8.159.2 Member Enumeration Documentation

#### 8.159.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
-----------	---



gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>This algorithm is well-suited for numerical matrices but generally suffers from the expensive division (and computation of GCDs) at each step.</p>
divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>This algorithm is only there for the purpose of cross-checks. It suffers from exponential intermediate expression swell. Use it only for small systems.</p>
bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. This is generally the fastest algorithm for solving linear systems. In contrast to division-free elimination it only has a linear expression swell. For two-dimensional systems, the two algorithms are equivalent, however.</p>
markowitz	<p>Markowitz-ordered Gaussian elimination. Same as the usual Gaussian elimination, but with additional effort spent on selecting pivots that minimize fill-in. Faster than the methods above for large sparse matrices (particularly with symbolic coefficients), otherwise slightly slower than Gaussian elimination.</p>

The documentation for this class was generated from the following file:

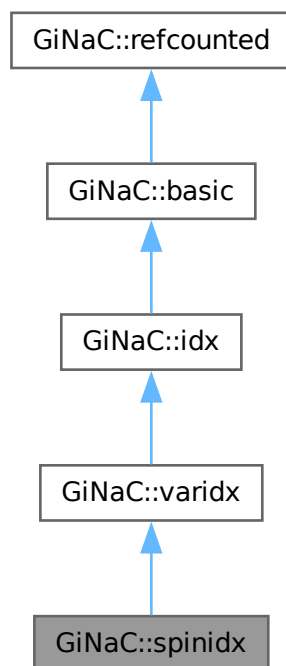
- [flags.h](#)

## 8.160 GiNaC::spinidx Class Reference

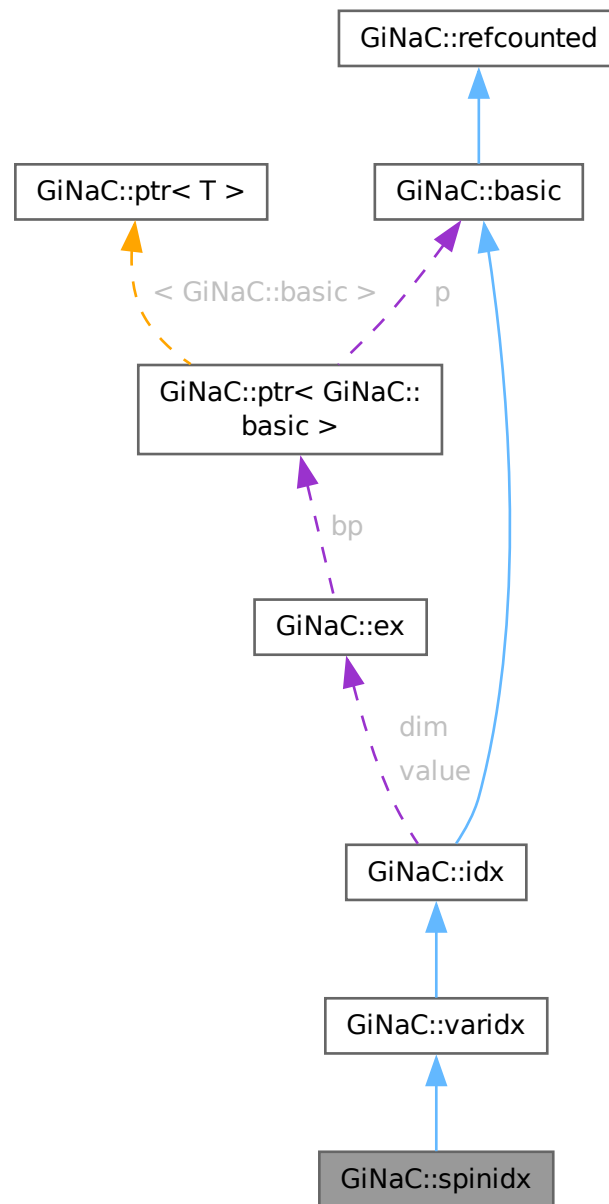
This class holds a spinor index that can be dotted or undotted and that also has a variance.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::spinidx:



Collaboration diagram for GiNaC::spinidx:



## Public Member Functions

- `spinidx` (const `ex` &`v`, const `ex` &`dim`=2, bool `covariant`=false, bool `dotted`=false)  
Construct index with given value, dimension, variance and dot.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override

- *Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override
- *Load (deserialize) the object from an archive node.*
- bool `is_dotted` () const
- *Check whether the index is dotted.*
- bool `is_undotted` () const
- *Check whether the index is not dotted.*
- `ex toggle_dot` () const
- *Make a new index with the same value and variance but the opposite dottedness.*
- `ex toggle_variance_dot` () const
- *Make a new index with the same value but opposite variance and dottedness.*

## Public Member Functions inherited from `GiNaC::varidx`

- `varidx` (const `ex` &v, const `ex` &dim, bool covariant=false)
- *Construct index with given value, dimension and variance.*
- bool `is_dummy_pair_same_type` (const `basic` &other) const override
- *Check whether the index forms a dummy index pair with another index of the same type.*
- void `archive` (`archive_node` &n) const override
- *Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override
- *Load (deserialize) the object from an archive node.*
- bool `is_covariant` () const
- *Check whether the index is covariant.*
- bool `is_contravariant` () const
- *Check whether the index is contravariant (not covariant).*
- `ex toggle_variance` () const
- *Make a new index with the same value but the opposite variance.*

## Public Member Functions inherited from `GiNaC::idx`

- `idx` (const `ex` &v, const `ex` &dim)
- *Construct index with given value and dimension.*
- bool `info` (unsigned inf) const override
- *Information about the object.*
- `size_t nops` () const override
- *Number of operands/members.*
- `ex op` (size\_t i) const override
- *Return operand/member at position i.*
- `ex map` (`map_function` &f) const override
- *Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf` () const override
- *By default, `basic::evalf` would evaluate the index value but we don't want a.1 to become a.*
- `ex subs` (const `exmap` &m, unsigned options=0) const override
- *Substitute a set of objects by arbitrary expressions.*
- `ex get_value` () const
- *Get value of index.*
- bool `is_numeric` () const
- *Check whether the index is numeric.*

- bool `is_symbolic` () const  
*Check whether the index is symbolic.*
- `ex get_dim` () const  
*Get dimension of index space.*
- bool `is_dim_numeric` () const  
*Check whether the dimension is numeric.*
- bool `is_dim_symbolic` () const  
*Check whether the dimension is symbolic.*
- `ex replace_dim` (const `ex` &new\_dim) const  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim` (const `idx` &other) const  
*Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

- Check whether this is a polynomial in the given variables.*

  - virtual int [degree](#) (const [ex](#) &s) const

*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const

*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const

*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const

*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const

*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const

*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const

*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const

*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const

*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>
- void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const

*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const

*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const

*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const

*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const

*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const

*Stop further evaluation.*
- unsigned [gethash](#) () const

- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::varidx](#)

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::idx](#)

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an index always returns 0.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_index](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- bool `dotted`

## Protected Attributes inherited from `GiNaC::varidx`

- bool `covariant`  
 *$x.mu$ , default is contravariant:  $x \sim \mu$*

## Protected Attributes inherited from `GiNaC::idx`

- `ex` value  
*Expression that constitutes the index (numeric or symbolic name).*
- `ex` `dim`  
*Dimension of space (can be symbolic or numeric).*

## Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 8.160.1 Detailed Description

This class holds a spinor index that can be dotted or undotted and that also has a variance.

This is used in the Weyl-van-der-Waerden formalism where the dot indicates complex conjugation. There is an associated (asymmetric) metric tensor that can be used to raise/lower spinor indices.

### 8.160.2 Constructor & Destructor Documentation

#### 8.160.2.1 `spinidx()`

```
GiNaC::spinidx::spinidx (
    const ex & v,
    const ex & dim = 2,
    bool covariant = false,
    bool dotted = false)
```

Construct index with given value, dimension, variance and dot.

#### Parameters

<code>v</code>	Value of index (numeric or symbolic)
<code>dim</code>	Dimension of index space (numeric or symbolic)



<a href="#"><i>covariant</i></a>	Make covariant index (default is contravariant)
<a href="#"><i>dotted</i></a>	Make covariant dotted (default is undotted)

References [dotted](#).

Referenced by [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [toggle\\_dot\(\)](#), and [toggle\\_variance\\_dot\(\)](#).

## 8.160.3 Member Function Documentation

### 8.160.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::spinidx::is_dummy_pair_same_type (
    const basic & other) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

References [dotted](#), and [spinidx\(\)](#).

### 8.160.3.2 conjugate()

```
ex GiNaC::spinidx::conjugate () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [toggle\\_dot\(\)](#).

### 8.160.3.3 archive()

```
void GiNaC::spinidx::archive (
    archive\_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

References [GiNaC::archive\\_node::add\\_bool\(\)](#), and [dotted](#).

### 8.160.3.4 read\_archive()

```
void GiNaC::spinidx::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

References [dotted](#), and [GiNaC::archive\\_node::find\\_bool\(\)](#).

### 8.160.3.5 match\_same\_type()

```
bool GiNaC::spinidx::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

References [dotted](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [spinidx\(\)](#).

### 8.160.3.6 is\_dotted()

```
bool GiNaC::spinidx::is_dotted () const [inline]
```

Check whether the index is dotted.

References [dotted](#).

**8.160.3.7 is\_undotted()**

```
bool GiNaC::spinidx::is_undotted () const [inline]
```

Check whether the index is not dotted.

References [dotted](#).

**8.160.3.8 toggle\_dot()**

```
ex GiNaC::spinidx::toggle_dot () const
```

Make a new index with the same value and variance but the opposite dottedness.

References [GiNaC::basic::clearflag\(\)](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), and [spinidx\(\)](#).

Referenced by [conjugate\(\)](#).

**8.160.3.9 toggle\_variance\_dot()**

```
ex GiNaC::spinidx::toggle_variance_dot () const
```

Make a new index with the same value but opposite variance and dottedness.

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::varidx::covariant](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), and [spinidx\(\)](#).

**8.160.3.10 do\_print()**

```
void GiNaC::spinidx::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [GiNaC::varidx::covariant](#), [dotted](#), [GiNaC::idx::print\\_index\(\)](#), and [GiNaC::print\\_context::s](#).

**8.160.3.11 do\_print\_latex()**

```
void GiNaC::spinidx::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

References [dotted](#), [GiNaC::idx::print\\_index\(\)](#), and [GiNaC::print\\_context::s](#).

**8.160.3.12 do\_print\_tree()**

```
void GiNaC::spinidx::do_print_tree (
    const print_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::varidx::covariant](#), [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::idx::dim](#), [dotted](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::print\\_context::s](#), and [GiNaC::idx::value](#).

## 8.160.4 Member Data Documentation

### 8.160.4.1 dotted

```
bool GiNaC::spinidx::dotted [protected]
```

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [is\\_dotted\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_undotted\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [spinidx\(\)](#), [toggle\\_dot\(\)](#), and [toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

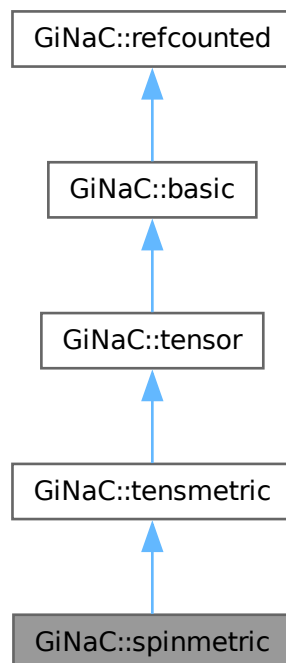
- [idx.h](#)
- [idx.cpp](#)

## 8.161 GiNaC::spinmetric Class Reference

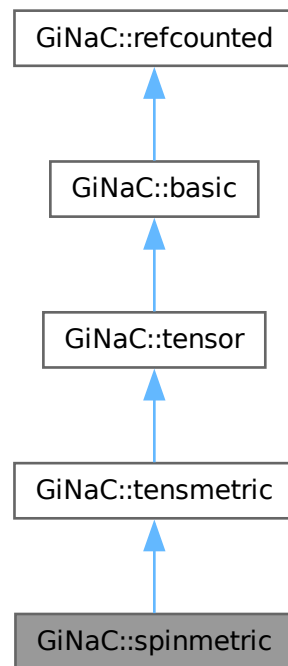
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::spinmetric:



Collaboration diagram for GiNaC::spinmetric:



### Public Member Functions

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed spinor metric with something else.*

### Public Member Functions inherited from `GiNaC::tensmetric`

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprnttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

- Return coefficient of degree  $n$  in object  $s$ .*

    - virtual `ex expand` (unsigned options=0) const

*Expand expression, i.e.*
  - virtual `ex collect` (const `ex` & $s$ , bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
  - virtual `ex series` (const `relational` & $r$ , int order, unsigned options=0) const

*Default implementation of `ex::series()`.*
  - virtual `ex normal` (`exmap` & $repl$ , `exmap` & $rev\_lookup$ , `lst` & $modifier$ ) const

*Default implementation of `ex::normal()`.*
  - virtual `ex to_rational` (`exmap` & $repl$ ) const

*Default implementation of `ex::to_rational()`.*
  - virtual `ex to_polynomial` (`exmap` & $repl$ ) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` & $xi$ ) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` & $self$ , const `ex` & $other$ ) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` & $self$ , const `numeric` & $other$ ) const
- Multiply an indexed expression with a scalar.*
- virtual `return_type_t return_type_info` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>
  - void `print_dispatch` (const `print_context` & $c$ , unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` & $ri$ , const `print_context` & $c$ , unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` & $n$ ) const
- Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` & $n$ , `lst` & $syms$ )
- Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` & $m$ , unsigned options) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` & $s$ , unsigned  $nth$ =1) const
- Default interface of  $n$ th derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` & $other$ ) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` & $other$ ) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned  $f$ ) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned  $f$ ) const
- Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensmetric](#)

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*



## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 8.161.1 Detailed Description

This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

If indexed, it must have exactly two indices of the same type which must be of class `spinidx` or a subclass and have dimension 2.

### 8.161.2 Member Function Documentation

#### 8.161.2.1 `info()`

```
bool GiNaC::spinmetric::info (
    unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

#### 8.161.2.2 `eval_indexed()`

```
ex GiNaC::spinmetric::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), and [GiNaC::basic::op\(\)](#).

### 8.161.2.3 contract\_with()

```
bool GiNaC::spinmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed spinor metric with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex\\_2](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::varidx::is\\_covariant\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::idx::is\\_symbolic\(\)](#).

### 8.161.2.4 do\_print()

```
void GiNaC::spinmetric::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.161.2.5 do\_print\_latex()

```
void GiNaC::spinmetric::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

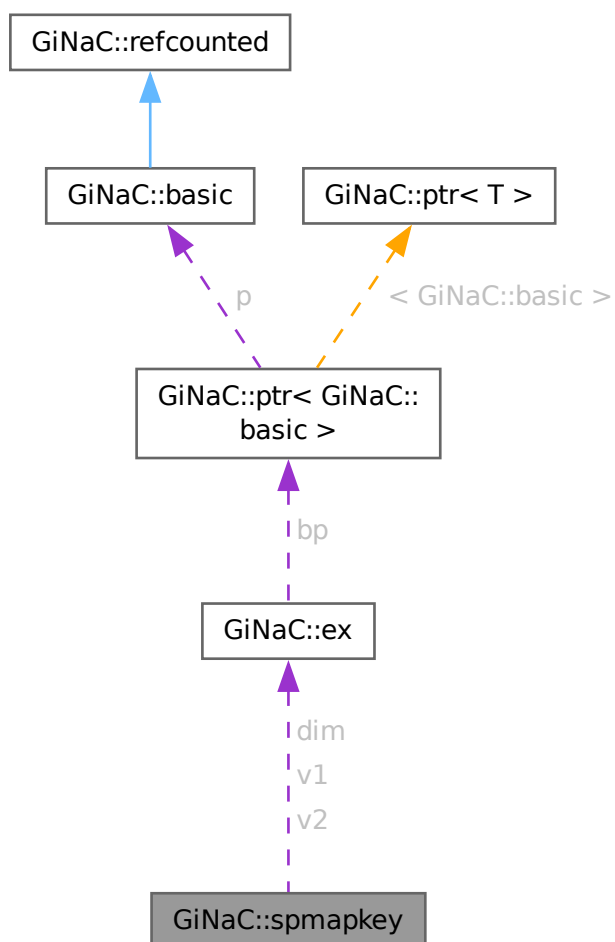
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 8.162 GiNaC::spmapkey Class Reference

```
#include <indexed.h>
```

Collaboration diagram for GiNaC::spmapkey:



### Public Member Functions

- `spmapkey ()`
- `spmapkey (const ex &v1, const ex &v2, const ex &dim=wild())`
- `bool operator== (const spmapkey &other) const`
- `bool operator< (const spmapkey &other) const`
- `void debugprint () const`

### Protected Attributes

- `ex v1`
- `ex v2`
- `ex dim`

## 8.162.1 Constructor & Destructor Documentation

### 8.162.1.1 `spmapkey()` [1/2]

`GiNaC::spmapkey::spmapkey () [inline]`

References [dim](#), and [GiNaC::wild\(\)](#).

Referenced by [operator<\(\)](#), and [operator==\(\)](#).

### 8.162.1.2 `spmapkey()` [2/2]

```
GiNaC::spmapkey::spmapkey (
    const ex & v1,
    const ex & v2,
    const ex & dim = wild\(\))
```

References [GiNaC::ex::compare\(\)](#), [dim](#), [GiNaC::is\\_a\(\)](#), [GiNaC::ex::op\(\)](#), [v1](#), and [v2](#).

## 8.162.2 Member Function Documentation

### 8.162.2.1 `operator==()`

```
bool GiNaC::spmapkey::operator== (
    const spmapkey & other) const
```

References [dim](#), [GiNaC::is\\_a\(\)](#), [spmapkey\(\)](#), [v1](#), and [v2](#).

### 8.162.2.2 `operator<()`

```
bool GiNaC::spmapkey::operator< (
    const spmapkey & other) const
```

References [dim](#), [GiNaC::is\\_a\(\)](#), [spmapkey\(\)](#), [v1](#), and [v2](#).

### 8.162.2.3 `debugprint()`

```
void GiNaC::spmapkey::debugprint () const
```

References [dim](#), [v1](#), and [v2](#).

Referenced by [GiNaC::scalar\\_products::debugprint\(\)](#).

## 8.162.3 Member Data Documentation

### 8.162.3.1 `v1`

`ex GiNaC::spmapkey::v1 [protected]`

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).

### 8.162.3.2 v2

`ex` `GiNaC::spmapkey::v2` [protected]

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).

### 8.162.3.3 dim

`ex` `GiNaC::spmapkey::dim` [protected]

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), [spmapkey\(\)](#), and [spmapkey\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 8.163 GiNaC::status\_flags Class Reference

Flags to store information about the state of an object.

```
#include <flags.h>
```

### Public Types

- enum {  
[dynallocated](#) = 0x0001 , [evaluated](#) = 0x0002 , [expanded](#) = 0x0004 , [hash\\_calculated](#) = 0x0008 ,  
[not\\_shareable](#) = 0x0010 , [has\\_indices](#) = 0x0020 , [has\\_no\\_indices](#) = 0x0040 , [is\\_positive](#) = 0x0080 ,  
[is\\_negative](#) = 0x0100 , [purely\\_indefinite](#) = 0x0200 }

### 8.163.1 Detailed Description

Flags to store information about the state of an object.

See also

[basic::flags](#)

### 8.163.2 Member Enumeration Documentation

#### 8.163.2.1 anonymous enum

anonymous enum

#### Enumerator

<code>dynallocated</code>	heap-allocated (i.e. created by <code>new</code> if we want to be clever and bypass the stack,  See also  <a href="#">ex::construct_from_basic()</a> )
---------------------------	--

evaluated	.eval() has already done its job
expanded	.expand(0) has already done its job (other <a href="#">expand()</a> options ignore this flag)
hash_calculated	.calchash() has already done its job
not_shareable	don't share instances of this object between different expressions unless explicitly asked to (used by <a href="#">ex::compare()</a> )
has_indices	
has_no_indices	
is_positive	
is_negative	
purely_indefinite	

The documentation for this class was generated from the following file:

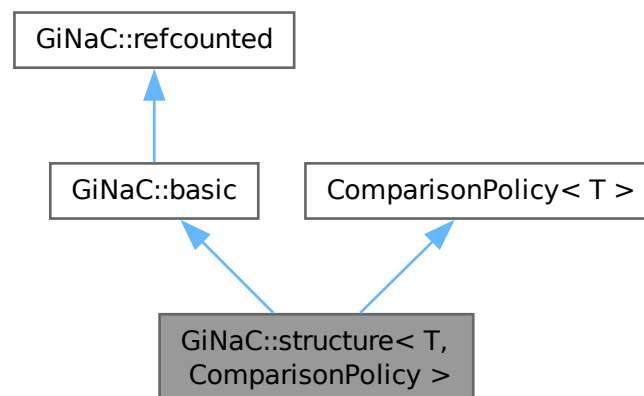
- [flags.h](#)

## 8.164 GiNaC::structure< T, ComparisonPolicy > Class Template Reference

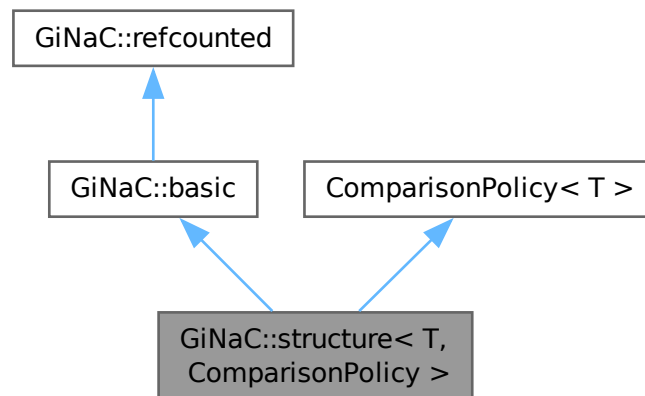
Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include <structure.h>
```

Inheritance diagram for GiNaC::structure< T, ComparisonPolicy >:



Collaboration diagram for GiNaC::structure< T, ComparisonPolicy >:



## Public Member Functions

- `structure` (const T &t)  
*Construct structure as a copy of a given C++ structure.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex eval_indexed` (const `basic` &i) const override  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- void `print` (const `print_context` &c, unsigned level=0) const override  
*Output to stream.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex operator[]` (const `ex` &index) const override
- `ex operator[]` (size\_t i) const override
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex & operator[]` (const `ex` &index) override
- `ex & operator[]` (size\_t i) override
- bool `has` (const `ex` &other, unsigned options=0) const override  
*Test for occurrence of a pattern.*
- bool `match` (const `ex` &pattern, `exmap` &repl\_list) const override  
*Check whether the expression matches a given pattern.*

- `ex subs` (const `exmap` &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex coeff` (const `ex` &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- `ex expand` (unsigned options=0) const override  
*Expand expression, i.e.*
- `ex collect` (const `ex` &s, bool distributed=false) const override  
*Sort expanded expression in terms of powers of some object(s).*
- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Default implementation of `ex::normal()`.*
- `ex to_rational` (`exmap` &repl) const override  
*Default implementation of `ex::to_rational()`.*
- `ex to_polynomial` (`exmap` &repl) const override
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex add_indexed` (const `ex` &self, const `ex` &other) const override  
*Add two indexed expressions.*
- `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const override  
*Multiply an indexed expression with a scalar.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Try to contract two indexed expressions that appear in the same product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- const T \* `operator->` () const
- T & `get_struct` ()
- const T & `get_struct` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalf` () const



- *Evaluate object numerically.*  
virtual [ex eval\\_integ](#) () const
- *Evaluate integrals, if result is known.*  
virtual void [dbgprint](#) () const
- *Little wrapper around print to be called within a debugger.*  
virtual void [dbgprinttree](#) () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual void [accept](#) (GiNaC::visitor &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override
- bool `match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex derivative` (const `symbol` &`s`) const override  
*Default implementation of `ex::diff()`.*
- bool `is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual int `compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &`c`, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const  
*Python parsable output to stream.*

### Static Private Member Functions

- static const char \* `get_class_name` ()

### Private Attributes

- T `obj`

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 8.164.1 Detailed Description

```
template<class T, template< class > class ComparisonPolicy>
class GiNaC::structure< T, ComparisonPolicy >
```

Wrapper template for making `GiNaC` classes out of C++ structures.

## 8.164.2 Constructor & Destructor Documentation

### 8.164.2.1 structure()

```
template<class T, template< class > class ComparisonPolicy>
GiNaC::structure< T, CP >::structure (
    const T & t) [inline]
```

Construct structure as a copy of a given C++ structure.

Default constructor.

References [obj](#).

Referenced by [is\\_equal\\_same\\_type\(\)](#).

## 8.164.3 Member Function Documentation

### 8.164.3.1 get\_class\_name()

```
template<class T, template< class > class ComparisonPolicy>
const char * GiNaC::structure< T, ComparisonPolicy >::get_class_name () [inline], [static],
[private]
```

References [get\\_class\\_name\(\)](#).

Referenced by [get\\_class\\_name\(\)](#).

### 8.164.3.2 eval()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval () const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::basic::hold\(\)](#).

### 8.164.3.3 evalm()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::evalm () const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.4 eval\_ncmul()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_ncmul (
    const exvector & v) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::hold\\_ncmul\(\)](#).

**8.164.3.5 eval\_indexed()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_indexed (
    const basic & i) const [inline], [override], [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [GiNaC::basic::ex](#), and [GiNaC::basic::hold\(\)](#).

**8.164.3.6 print()**

```
template<class T, template< class > class ComparisonPolicy>
void GiNaC::structure< T, ComparisonPolicy >::print (
    const print_context & c,
    unsigned level = 0) const [inline], [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

**Parameters**

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

**8.164.3.7 precedence()**

```
template<class T, template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::precedence () const [inline], [override],
[virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

**8.164.3.8 info()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::info (
    unsigned int) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

**8.164.3.9 nops()**

```
template<class T, template< class > class ComparisonPolicy>
size_t GiNaC::structure< T, ComparisonPolicy >::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

**8.164.3.10 op()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::op (
    size_t i) const [inline], [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.11 operator[]() [1/4]**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.12 operator[]() [2/4]**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.13 let\_op()**

```
template<class T, template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::let_op (
    size_t i) [inline], [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.14 operator[]() [3/4]**

```
template<class T, template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.15 operator[]() [4/4]**

```
template<class T, template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.16 has()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::has (
    const ex & pattern,
    unsigned options = 0) const [inline], [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.17 match()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match (
    const ex & pattern,
    exmap & repl_lst) const [inline], [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.18 match\_same\_type()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match_same_type (
    const basic & other) const [inline], [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#).

**8.164.3.19 subs()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::subs (
    const exmap & m,
    unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The *ex* returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

#### 8.164.3.20 map()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::map (
    map_function & f) const [inline], [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

#### 8.164.3.21 degree()

```
template<class T, template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::degree (
    const ex & s) const [inline], [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

#### 8.164.3.22 ldegree()

```
template<class T, template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::ldegree (
    const ex & s) const [inline], [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

#### 8.164.3.23 coeff()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::coeff (
    const ex & s,
    int n = 1) const [inline], [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).



**8.164.3.24 expand()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::expand (
    unsigned options = 0) const [inline], [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.25 collect()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::collect (
    const ex & s,
    bool distributed = false) const [inline], [override], [virtual]
```

Sort expanded expression in terms of powers of some object(s).

**Parameters**

<i>s</i>	object(s) to sort in
<i>distributed</i>	recursive or distributed form (only used when s is a list)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.26 derivative()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::derivative (
    const symbol & s) const [inline], [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.27 series()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::series (
    const relational & r,
    int order,
    unsigned options = 0) const [inline], [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.28 normal()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [inline], [override], [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.29 to\_rational()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_rational (
    exmap & repl) const [inline], [override], [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.30 to\_polynomial()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_polynomial (
    exmap & repl) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.31 integer\_content()

```
template<class T, template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::integer_content () const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

### 8.164.3.32 smod()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::smod (
    const numeric & xi) const [inline], [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

#### Parameters

$xi$	modulus
------	---------

#### Returns

mapped polynomial

#### See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.33 max\_coefficient()**

```
template<class T, template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::max_coefficient () const [inline], [override],
[virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

**8.164.3.34 get\_free\_indices()**

```
template<class T, template< class > class ComparisonPolicy>
exvector GiNaC::structure< T, ComparisonPolicy >::get_free_indices () const [inline], [override],
[virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

**8.164.3.35 add\_indexed()**

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::add_indexed (
    const ex & self,
    const ex & other) const [inline], [override], [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class indexed (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

**Parameters**

<i>self</i>	First indexed expression; its base object is *this
<i>other</i>	Second indexed expression

**Returns**

sum of self and other

See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.164.3.36 scalar\_mul\_indexed()

```
template<class T, template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::scalar_mul_indexed (
    const ex & self,
    const numeric & other) const [inline], [override], [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

#### Parameters

<i>self</i>	Indexed expression; its base object is *this
-------------	--

<i>other</i>	Numeric value
--------------	---------------

**Returns**

product of self and other

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.164.3.37 contract\_with()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [inline], [override], [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

**Parameters**

<i>self</i>	Pointer to first indexed expression; its base object is *this
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

**Returns**

true if the contraction was successful, false otherwise

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

**8.164.3.38 return\_type()**

```
template<class T, template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::return_type () const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

**8.164.3.39 return\_type\_tinfo()**

```
template<class T, template< class > class ComparisonPolicy>
return_type_t GiNaC::structure< T, ComparisonPolicy >::return_type_tinfo () const [inline],
[override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**8.164.3.40 is\_equal\_same\_type()**

```
template<class T, template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type (
    const basic & other) const [inline], [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [obj](#), and [structure\(\)](#).

**8.164.3.41 calchash()**

```
template<class T, template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::calchash () const [inline], [override],
[protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

#### 8.164.3.42 operator->()

```
template<class T, template< class > class ComparisonPolicy>
const T * GiNaC::structure< T, ComparisonPolicy >::operator-> () const [inline]
```

References [obj](#).

#### 8.164.3.43 get\_struct() [1/2]

```
template<class T, template< class > class ComparisonPolicy>
T & GiNaC::structure< T, ComparisonPolicy >::get_struct () [inline]
```

References [obj](#).

#### 8.164.3.44 get\_struct() [2/2]

```
template<class T, template< class > class ComparisonPolicy>
const T & GiNaC::structure< T, ComparisonPolicy >::get_struct () const [inline]
```

References [obj](#).

### 8.164.4 Member Data Documentation

#### 8.164.4.1 obj

```
template<class T, template< class > class ComparisonPolicy>
T GiNaC::structure< T, ComparisonPolicy >::obj [private]
```

Referenced by [get\\_struct\(\)](#), [get\\_struct\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [operator->\(\)](#), and [structure\(\)](#).

The documentation for this class was generated from the following file:

- [structure.h](#)

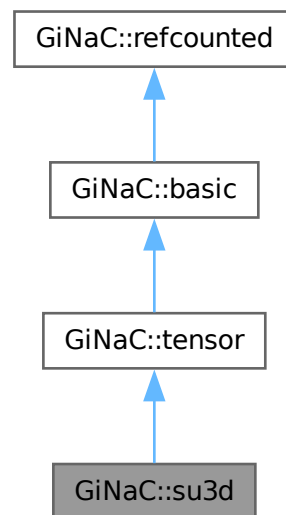


## 8.165 GiNaC::su3d Class Reference

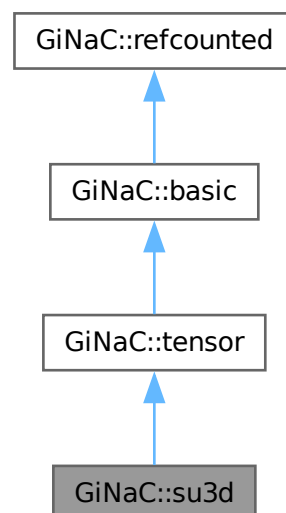
This class represents the tensor of symmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3d:



Collaboration diagram for GiNaC::su3d:



## Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed symmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed symmetric structure constant with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- `const basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () const  
*Create a clone of this object on the heap.*
- `virtual ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () const  
*Evaluate object numerically.*
- `virtual ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- `virtual void print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- `virtual void dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- `virtual bool info` (unsigned inf) const  
*Information about the object.*
- `virtual size_t nops` () const  
*Number of operands/members.*
- `virtual ex op` (size\_t i) const  
*Return operand/member at position i.*
- `virtual ex operator[]` (const `ex` &index) const
- `virtual ex operator[]` (size\_t i) const
- `virtual ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- `virtual ex & operator[]` (const `ex` &index)
- `virtual ex & operator[]` (size\_t i)

- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `unsigned return_type` () const override
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- `unsigned return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual bool match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.165.1 Detailed Description

This class represents the tensor of symmetric su(3) structure constants.

## 8.165.2 Member Function Documentation

### 8.165.2.1 [eval\\_indexed\(\)](#)

```
ex GiNaC::su3d::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of indexed symmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_6](#), [CMPINDICES](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 8.165.2.2 [contract\\_with\(\)](#)

```
bool GiNaC::su3d::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed symmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_from](#).

### 8.165.2.3 return\_type()

```
unsigned GiNaC::su3d::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.165.2.4 do\_print()

```
void GiNaC::su3d::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

### 8.165.2.5 do\_print\_latex()

```
void GiNaC::su3d::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

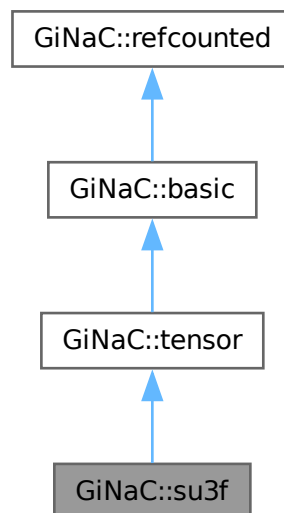
- [color.h](#)
- [color.cpp](#)

## 8.166 GiNaC::su3f Class Reference

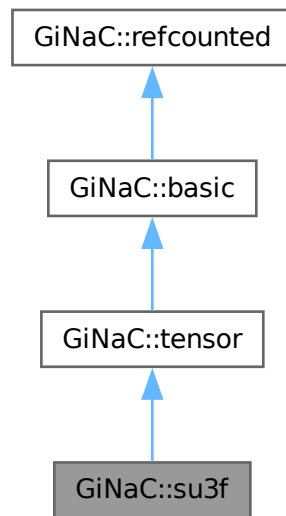
This class represents the tensor of antisymmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3f:



Collaboration diagram for GiNaC::su3f:



### Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed antisymmetric structure constant.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed antisymmetric structure constant with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*

- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual size\_t `nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*



- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.166.1 Detailed Description

This class represents the tensor of antisymmetric su(3) structure constants.

## 8.166.2 Member Function Documentation

### 8.166.2.1 eval\_indexed()

```
ex GiNaC::su3f::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of indexed antisymmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [CMPINDICES](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 8.166.2.2 contract\_with()

```
bool GiNaC::su3f::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed antisymmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::I](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 8.166.2.3 return\_type()

```
unsigned GiNaC::su3f::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.166.2.4 do\_print()

```
void GiNaC::su3f::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.166.2.5 do\_print\_latex()

```
void GiNaC::su3f::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

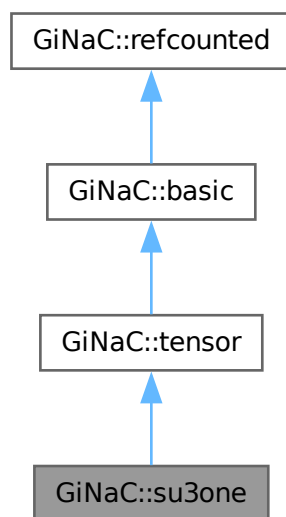
- [color.h](#)
- [color.cpp](#)

## 8.167 GiNaC::su3one Class Reference

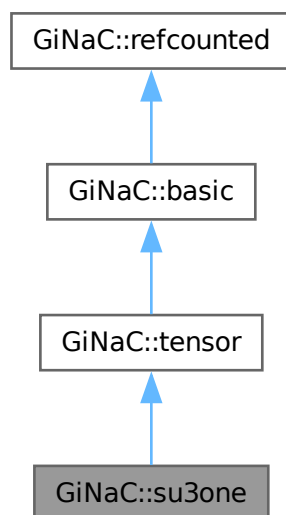
This class represents the  $su(3)$  unity element.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3one:



Collaboration diagram for GiNaC::su3one:



**Protected Member Functions**

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

**Protected Member Functions inherited from `GiNaC::tensor`**

- unsigned `return_type` () const override

**Protected Member Functions inherited from `GiNaC::basic`**

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Additional Inherited Members****Public Member Functions inherited from `GiNaC::tensor`**

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- const `basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual size\_t `nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int [ldegree](#) (const [ex](#) &s) const
- Return degree of lowest power in object s.*

  - virtual [ex coeff](#) (const [ex](#) &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual [ex expand](#) (unsigned options=0) const
- Expand expression, i.e.*

  - virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const
- Default implementation of [ex::series\(\)](#).*

  - virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const
- Default implementation of [ex::normal\(\)](#).*

  - virtual [ex to\\_rational](#) ([exmap](#) &repl) const
- Default implementation of [ex::to\\_rational\(\)](#).*

  - virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual [numeric max\\_coefficient](#) () const
- Implementation [ex::max\\_coefficient\(\)](#).*

  - virtual [exvector get\\_free\\_indices](#) () const
- Return a vector containing the free indices of an expression.*

  - virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const
- Add two indexed expressions.*

  - virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>
  - void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const
- Like [print\(\)](#), but dispatch to the specified class.*

  - void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const
- Like [print\(\)](#), but dispatch to the specified class.*

  - virtual void [archive](#) ([archive\\_node](#) &n) const
- Save (serialize) the object into archive node.*

  - virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)
- Load (deserialize) the object from an archive node.*

  - [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const
- Helper function for [subs\(\)](#).*

  - [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const
- Default interface of nth derivative [ex::diff\(s, n\)](#).*

  - int [compare](#) (const [basic](#) &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool [is\\_equal](#) (const [basic](#) &other) const
- Test for syntactic equality.*

  - const [basic](#) & [hold](#) () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const

*Set some [status\\_flags](#).*

- const [basic](#) & [clearflag](#) (unsigned f) const

*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.167.1 Detailed Description

This class represents the su(3) unity element.

### 8.167.2 Member Function Documentation

#### 8.167.2.1 [do\\_print\(\)](#)

```
void GiNaC::su3one::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

#### 8.167.2.2 [do\\_print\\_latex\(\)](#)

```
void GiNaC::su3one::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following file:

- [color.h](#)

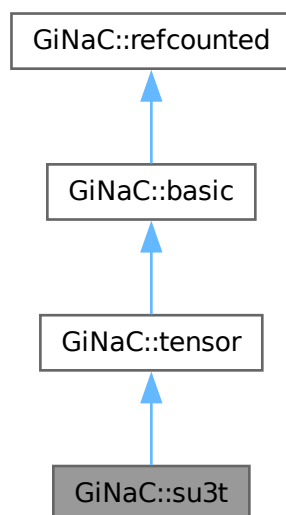


## 8.168 GiNaC::su3t Class Reference

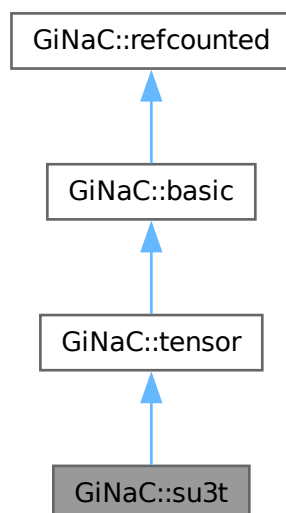
This class represents an  $su(3)$  generator.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3t:



Collaboration diagram for GiNaC::su3t:



## Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of generator with something else.*

## Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)

- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- `unsigned return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual bool match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void ensure_if_modifiable` () const

*Ensure the object may be modified without hurting others, throws if this is not the case.*

- void `do_print` (const `print_context` &c, unsigned level) const

*Default output to stream.*

- void `do_print_tree` (const `print_tree` &c, unsigned level) const

*Tree output to stream.*

- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

*Python parsable output to stream.*

## Additional Inherited Members

## Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 8.168.1 Detailed Description

This class represents an su(3) generator.

### 8.168.2 Member Function Documentation

#### 8.168.2.1 `contract_with()`

```
bool GiNaC::su3t::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of generator with something else.

Reimplemented from `GiNaC::basic`.

References `GiNaC::ex1`, `GiNaC::color_ONE()`, `GiNaC::color_trace()`, `GiNaC::ex_to()`, `GINAC_ASSERT`, `GiNaC::is_a()`, and `GiNaC::is_exactly_a()`.

#### 8.168.2.2 `do_print()`

```
void GiNaC::su3t::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.168.2.3 do\_print\_latex()

```
void GiNaC::su3t::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 8.169 GiNaC::subs\_options Class Reference

Flags to control the behavior of [subs\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum {  
[no\\_pattern](#) = 0x0001 , [subs\\_no\\_pattern](#) = 0x0001 , [algebraic](#) = 0x0002 , [subs\\_algebraic](#) = 0x0002 ,  
[pattern\\_is\\_product](#) = 0x0004 , [pattern\\_is\\_not\\_product](#) = 0x0008 , [no\\_index\\_renaming](#) = 0x0010 ,  
[really\\_subs\\_idx](#) = 0x0020 }

### 8.169.1 Detailed Description

Flags to control the behavior of [subs\(\)](#).

### 8.169.2 Member Enumeration Documentation

#### 8.169.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

<a href="#">no_pattern</a>	disable pattern matching
<a href="#">subs_no_pattern</a>	
<a href="#">algebraic</a>	enable algebraic substitutions
<a href="#">subs_algebraic</a>	
<a href="#">pattern_is_product</a>	used internally by <a href="#">expairseq::subschildren()</a>
<a href="#">pattern_is_not_product</a>	used internally by <a href="#">expairseq::subschildren()</a>
<a href="#">no_index_renaming</a>	
<a href="#">really_subs_idx</a>	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 8.170 GiNaC::sy\_is\_less Class Reference

### Public Member Functions

- [sy\\_is\\_less](#) (exvector::iterator v\_)
- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### Private Attributes

- exvector::iterator [v](#)

### 8.170.1 Constructor & Destructor Documentation

#### 8.170.1.1 sy\_is\_less()

```
GiNaC::sy_is_less::sy_is_less (
    exvector::iterator v_) [inline]
```

References [v](#).

### 8.170.2 Member Function Documentation

#### 8.170.2.1 operator>()()

```
bool GiNaC::sy_is_less::operator() (
    const ex & lh,
    const ex & rh) const [inline]
```

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), and [v](#).

### 8.170.3 Member Data Documentation

#### 8.170.3.1 v

```
exvector::iterator GiNaC::sy_is_less::v [private]
```

Referenced by [operator>\(\)\(\)](#), and [sy\\_is\\_less\(\)](#).

The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 8.171 GiNaC::sy\_swap Class Reference

### Public Member Functions

- [sy\\_swap](#) (exvector::iterator v\_, bool &s)
- void [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh)

## Public Attributes

- `bool` & [swapped](#)

## Private Attributes

- `exvector::iterator` [v](#)

## 8.171.1 Constructor & Destructor Documentation

### 8.171.1.1 `sy_swap()`

```
GiNaC::sy_swap::sy_swap (  
    exvector::iterator v_,  
    bool & s) [inline]
```

References [swapped](#), and [v](#).

## 8.171.2 Member Function Documentation

### 8.171.2.1 `operator>()()`

```
void GiNaC::sy_swap::operator() (  
    const ex & lh,  
    const ex & rh) [inline]
```

References [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [swapped](#), and [v](#).

## 8.171.3 Member Data Documentation

### 8.171.3.1 `v`

```
exvector::iterator GiNaC::sy_swap::v [private]
```

Referenced by [operator>\(\)\(\)](#), and [sy\\_swap\(\)](#).

### 8.171.3.2 `swapped`

```
bool& GiNaC::sy_swap::swapped
```

Referenced by [operator>\(\)\(\)](#), and [sy\\_swap\(\)](#).

The documentation for this class was generated from the following file:

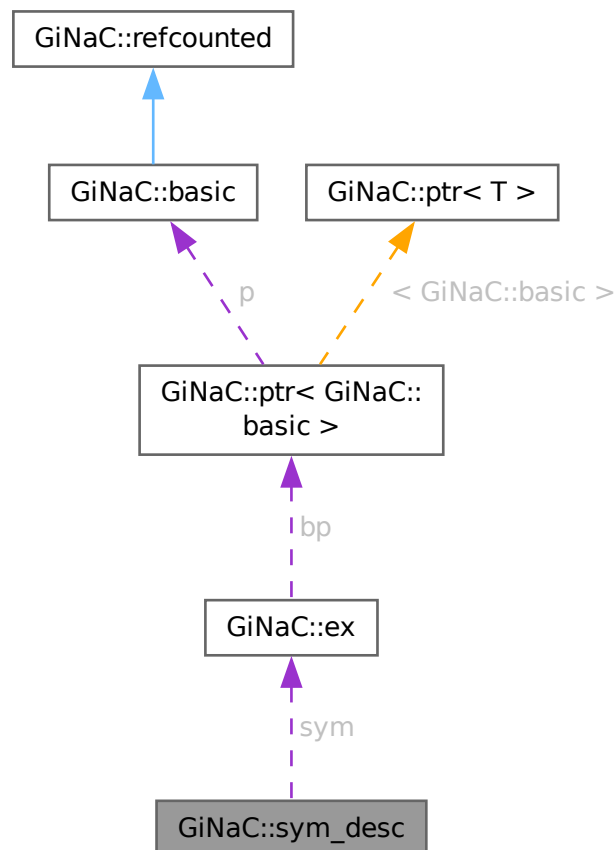
- [symmetry.cpp](#)



## 8.172 GiNaC::sym\_desc Struct Reference

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

Collaboration diagram for GiNaC::sym\_desc:



### Public Member Functions

- **sym\_desc** (const **ex** &s)  
*Initialize symbol, leave other variables uninitialized.*
- bool **operator<** (const **sym\_desc** &x) const  
*Comparison operator for sorting.*

### Public Attributes

- **ex sym**  
*Reference to symbol.*
- int **deg\_a**

- *Highest degree of symbol in polynomial "a".*  
• int [deg\\_b](#)
- *Highest degree of symbol in polynomial "b".*  
• int [ldeg\\_a](#)
- *Lowest degree of symbol in polynomial "a".*  
• int [ldeg\\_b](#)
- *Lowest degree of symbol in polynomial "b".*  
• int [max\\_deg](#)
- *Maximum of deg\_a and deg\_b (Used for sorting).*  
• size\_t [max\\_lcnops](#)
- *Maximum number of terms of leading coefficient of symbol in both polynomials.*

### 8.172.1 Detailed Description

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

A vector of these structures with information about all symbols in two polynomials can be created with the function [get\\_symbol\\_stats\(\)](#).

See also

[get\\_symbol\\_stats](#)

### 8.172.2 Constructor & Destructor Documentation

#### 8.172.2.1 sym\_desc()

```
GiNaC::sym_desc::sym_desc (
    const ex & s) [inline]
```

Initialize symbol, leave other variables uninitialized.

References [deg\\_a](#), [deg\\_b](#), [ldeg\\_a](#), [ldeg\\_b](#), [max\\_deg](#), [max\\_lcnops](#), and [sym](#).

Referenced by [operator<\(\)](#).

### 8.172.3 Member Function Documentation

#### 8.172.3.1 operator<()

```
bool GiNaC::sym_desc::operator< (
    const sym_desc & x) const [inline]
```

Comparison operator for sorting.

References [max\\_deg](#), [max\\_lcnops](#), and [sym\\_desc\(\)](#).

## 8.172.4 Member Data Documentation

### 8.172.4.1 sym

```
ex GiNaC::sym_desc::sym
```

Reference to symbol.

Referenced by [sym\\_desc\(\)](#).

### 8.172.4.2 deg\_a

```
int GiNaC::sym_desc::deg_a
```

Highest degree of symbol in polynomial "a".

Referenced by [sym\\_desc\(\)](#).

### 8.172.4.3 deg\_b

```
int GiNaC::sym_desc::deg_b
```

Highest degree of symbol in polynomial "b".

Referenced by [sym\\_desc\(\)](#).

### 8.172.4.4 ldeg\_a

```
int GiNaC::sym_desc::ldeg_a
```

Lowest degree of symbol in polynomial "a".

Referenced by [sym\\_desc\(\)](#).

### 8.172.4.5 ldeg\_b

```
int GiNaC::sym_desc::ldeg_b
```

Lowest degree of symbol in polynomial "b".

Referenced by [sym\\_desc\(\)](#).

### 8.172.4.6 max\_deg

```
int GiNaC::sym_desc::max_deg
```

Maximum of deg\_a and deg\_b (Used for sorting).

Referenced by [operator<\(\)](#), and [sym\\_desc\(\)](#).

#### 8.172.4.7 max\_lcnops

```
size_t GiNaC::sym_desc::max_lcnops
```

Maximum number of terms of leading coefficient of symbol in both polynomials.

Referenced by [operator<\(\)](#), and [sym\\_desc\(\)](#).

The documentation for this struct was generated from the following file:

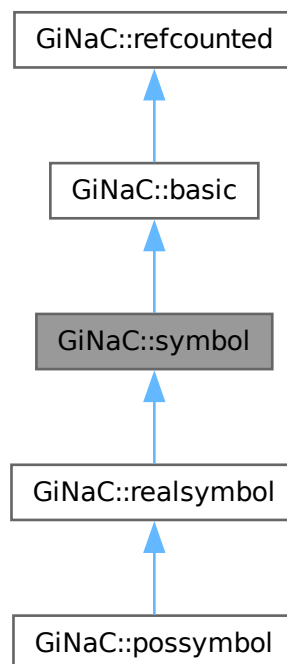
- [normal.cpp](#)

### 8.173 GiNaC::symbol Class Reference

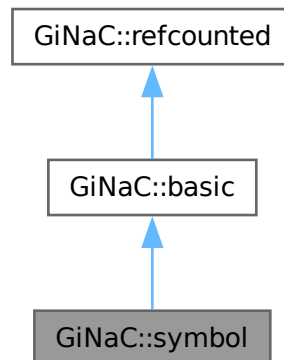
Basic CAS symbol.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::symbol:



Collaboration diagram for GiNaC::symbol:



### Public Member Functions

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int order, unsigned options=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const

- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printrname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Static Private Attributes

- static unsigned [next\\_serial](#) = 0

### 8.173.1 Detailed Description

Basic CAS symbol.

It has a name because it must know how to output itself.



## 8.173.2 Constructor & Destructor Documentation

### 8.173.2.1 symbol() [1/2]

```
GiNaC::symbol::symbol (
    const std::string & initname) [explicit]
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

Referenced by [derivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::realsymbol::realsymbol\(\)](#), [GiNaC::realsymbol::realsymbol\(\)](#), and [GiNaC::realsymbol::realsymbol\(\)](#).

### 8.173.2.2 symbol() [2/2]

```
GiNaC::symbol::symbol (
    const std::string & initname,
    const std::string & texname)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

## 8.173.3 Member Function Documentation

### 8.173.3.1 info()

```
bool GiNaC::symbol::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::expanded](#), [get\\_domain\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::domain::real](#), [GiNaC::info\\_flags::real](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), and [GiNaC::real\\_part\\_expl\\_derivative\(\)](#).

### 8.173.3.2 eval()

```
ex GiNaC::symbol::eval () const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.3 evalf()

```
ex GiNaC::symbol::evalf () const [inline], [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.4 series()

```
ex GiNaC::symbol::series (
    const relational & r,
    int order,
    unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for symbols.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::relational::lhs\(\)](#), and [GiNaC::relational::rhs\(\)](#).

### 8.173.3.5 subs()

```
ex GiNaC::symbol::subs (
    const exmap & m,
    unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 8.173.3.6 normal()

```
ex GiNaC::symbol::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for symbols.

This returns the unmodified symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::dynallocate\(\)](#), and [GiNaC::basic::ex](#).

### 8.173.3.7 to\_rational()

```
ex GiNaC::symbol::to_rational (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.8 to\_polynomial()

```
ex GiNaC::symbol::to_polynomial (
    exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.9 conjugate()

```
ex GiNaC::symbol::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.10 real\_part()

```
ex GiNaC::symbol::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

### 8.173.3.11 imag\_part()

```
ex GiNaC::symbol::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.173.3.12 is\_polynomial()**

```
bool GiNaC::symbol::is_polynomial (
    const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**8.173.3.13 archive()**

```
void GiNaC::symbol::archive (
    archive\_node & n) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_string\(\)](#), [name](#), and [TeX\\_name](#).

**8.173.3.14 read\_archive()**

```
void GiNaC::symbol::read_archive (
    const archive\_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

Read object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::append\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::is\\_a\(\)](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

**8.173.3.15 derivative()**

```
ex GiNaC::symbol::derivative (
    const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.

It returns 1 or 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::basic::compare\\_same\\_type\(\)](#), [GiNaC::basic::ex](#), and [symbol\(\)](#).

### 8.173.3.16 is\_equal\_same\_type()

```
bool GiNaC::symbol::is_equal_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::basic\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [serial](#), and [symbol\(\)](#).

Referenced by [series\(\)](#).

### 8.173.3.17 calchash()

```
unsigned GiNaC::symbol::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 8.173.3.18 get\_domain()

```
virtual unsigned GiNaC::symbol::get_domain () const [inline], [virtual]
```

Reimplemented in [GiNaC::possymbol](#), and [GiNaC::realsymbol](#).

References [GiNaC::domain::complex](#).

Referenced by [do\\_print\\_tree\(\)](#), and [info\(\)](#).

### 8.173.3.19 set\_name()

```
void GiNaC::symbol::set_name (
    const std::string & n) [inline]
```

References [name](#).

#### 8.173.3.20 `set_TeX_name()`

```
void GiNaC::symbol::set_TeX_name (
    const std::string & n) [inline]
```

References [TeX\\_name](#).

#### 8.173.3.21 `get_name()`

```
std::string GiNaC::symbol::get_name () const
```

References [name](#), and [serial](#).

Referenced by [do\\_print\(\)](#), and [get\\_TeX\\_name\(\)](#).

#### 8.173.3.22 `get_TeX_name()`

```
std::string GiNaC::symbol::get_TeX_name () const
```

References [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [get\\_name\(\)](#), and [TeX\\_name](#).

#### 8.173.3.23 `do_print()`

```
void GiNaC::symbol::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [get\\_name\(\)](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.173.3.24 `do_print_latex()`

```
void GiNaC::symbol::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

References [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [name](#), [GiNaC::print\\_context::s](#), [serial](#), and [TeX\\_name](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.173.3.25 `do_print_tree()`

```
void GiNaC::symbol::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::basic::flags](#), [get\\_domain\(\)](#), [GiNaC::basic::hashvalue](#), [name](#), [GiNaC::print\\_context::s](#), and [serial](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.173.3.26 do\_print\_python\_repr()

```
void GiNaC::symbol::do_print_python_repr (
    const print_python_repr & c,
    unsigned level) const [protected]
```

References [name](#), [GiNaC::print\\_context::s](#), [serial](#), and [TeX\\_name](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.173.4 Member Data Documentation

### 8.173.4.1 serial

```
unsigned GiNaC::symbol::serial [protected]
```

unique serial number for comparison

Referenced by [calchash\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [symbol\(\)](#), and [symbol\(\)](#).

### 8.173.4.2 name

```
std::string GiNaC::symbol::name [mutable], [protected]
```

printname of this symbol

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [read\\_archive\(\)](#), [set\\_name\(\)](#), [symbol\(\)](#), and [symbol\(\)](#).

### 8.173.4.3 TeX\_name

```
std::string GiNaC::symbol::TeX_name [protected]
```

LaTeX name of this symbol.

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [get\\_TeX\\_name\(\)](#), [read\\_archive\(\)](#), [set\\_TeX\\_name\(\)](#), [symbol\(\)](#), and [symbol\(\)](#).

### 8.173.4.4 next\_serial

```
unsigned GiNaC::symbol::next_serial = 0 [static], [private]
```

Referenced by [GiNaC::GINAC\\_BIND\\_UNARCHIVER\(\)](#), [read\\_archive\(\)](#), [symbol\(\)](#), and [symbol\(\)](#).

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symbol.cpp](#)

## 8.174 GiNaC::symbolset Class Reference

### Public Member Functions

- [symbolset](#) (const [ex](#) &e)
- [bool has](#) (const [ex](#) &e) const

### Private Member Functions

- void [insert\\_symbols](#) (const [ex](#) &e)

### Private Attributes

- [exset s](#)

## 8.174.1 Constructor & Destructor Documentation

### 8.174.1.1 symbolset()

```
GiNaC::symbolset::symbolset (  
    const ex & e) [inline], [explicit]
```

References [insert\\_symbols\(\)](#).

## 8.174.2 Member Function Documentation

### 8.174.2.1 insert\_symbols()

```
void GiNaC::symbolset::insert_symbols (  
    const ex & e) [inline], [private]
```

References [insert\\_symbols\(\)](#), [GiNaC::is\\_a\(\)](#), and [s](#).

Referenced by [insert\\_symbols\(\)](#), and [symbolset\(\)](#).

### 8.174.2.2 has()

```
bool GiNaC::symbolset::has (  
    const ex & e) const [inline]
```

References [s](#).

Referenced by [GiNaC::lsolve\(\)](#).



### 8.174.3 Member Data Documentation

#### 8.174.3.1 s

`exset` GiNaC::symbolset::s [private]

Referenced by [has\(\)](#), and [insert\\_symbols\(\)](#).

The documentation for this class was generated from the following file:

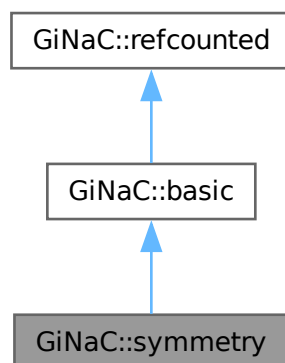
- [inifcns.cpp](#)

## 8.175 GiNaC::symmetry Class Reference

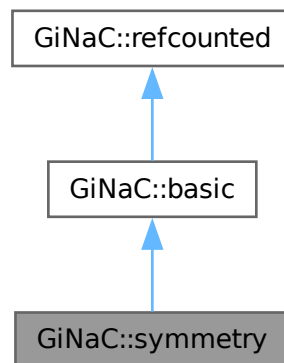
This class describes the symmetry of a group of indices.

```
#include <symmetry.h>
```

Inheritance diagram for GiNaC::symmetry:



Collaboration diagram for GiNaC::symmetry:



## Public Types

- enum [symmetry\\_type](#) { [none](#) , [symmetric](#) , [antisymmetric](#) , [cyclic](#) }
- Type of symmetry.*

## Public Member Functions

- [symmetry](#) (unsigned i)  
*Create leaf node that represents one index.*
- [symmetry](#) ([symmetry\\_type](#) t, const [symmetry](#) &c1, const [symmetry](#) &c2)  
*Create node with two children.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- [symmetry\\_type](#) [get\\_type](#) () const  
*Get symmetry type.*
- void [set\\_type](#) ([symmetry\\_type](#) t)  
*Set symmetry type.*
- [symmetry](#) & [add](#) (const [symmetry](#) &c)  
*Add child node, check index sets for consistency.*
- void [validate](#) (unsigned n)  
*Verify that all indices of this node are in the range [0..n-1].*
- bool [has\\_symmetry](#) () const  
*Check whether this node actually represents any kind of symmetry.*
- bool [has\\_nonsymmetric](#) () const  
*Check whether this node involves anything non symmetric.*
- bool [has\\_cyclic](#) () const  
*Check whether this node involves a cyclic symmetry.*

Public Member Functions inherited from **GiNaC::basic**

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex` `coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex` `expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex` `collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex` `series` (const `relational` &r, int order, unsigned options=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex` `normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex` `to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex` `to_polynomial` (`exmap` &repl) const
- virtual `numeric` `integer_content` () const
- virtual `ex` `smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric` `max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector` `get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex` `add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex` `scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t` `return_type_tinfo` () const
  - virtual `ex` `conjugate` () const
  - virtual `ex` `real_part` () const
  - virtual `ex` `imag_part` () const
- template<class T>
  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - `ex` `subs_one_level` (const `exmap` &m, unsigned options) const
- Helper function for `subs()`.*

  - `ex` `diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

  - unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*

  - const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- [symmetry\\_type](#) type  
*Type of symmetry described by this node.*
- std::set< unsigned > [indices](#)  
*Sorted union set of all indices handled by this node.*
- [exvector](#) children  
*Vector of child nodes.*

## Friends

- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- int [canonicalize](#) ([exvector::iterator](#) v, const [symmetry](#) &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 8.175.1 Detailed Description

This class describes the symmetry of a group of indices.

These objects can be grouped into a tree to form complex mixed symmetries.

## 8.175.2 Member Enumeration Documentation

### 8.175.2.1 [symmetry\\_type](#)

```
enum GiNaC::symmetry::symmetry\_type
```

Type of symmetry.

#### Enumerator

<a href="#">none</a>	no symmetry properties
<a href="#">symmetric</a>	totally symmetric
<a href="#">antisymmetric</a>	totally antisymmetric
<a href="#">cyclic</a>	cyclic symmetry

## 8.175.3 Constructor & Destructor Documentation

### 8.175.3.1 [symmetry\(\)](#) [1/2]

```
GiNaC::symmetry::symmetry (  
    unsigned i)
```

Create leaf node that represents one index.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [indices](#), [none](#), [GiNaC::basic::setflag\(\)](#), [symmetry\(\)](#), and [type](#).

Referenced by [add\(\)](#), [canonicalize](#), [sy\\_swap](#), [symmetry\(\)](#), and [symmetry\(\)](#).

### 8.175.3.2 symmetry() [2/2]

```
GiNaC::symmetry::symmetry (
    symmetry_type t,
    const symmetry & c1,
    const symmetry & c2)
```

Create node with two children.

References [add\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), [symmetry\(\)](#), and [type](#).

## 8.175.4 Member Function Documentation

### 8.175.4.1 archive()

```
void GiNaC::symmetry::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [children](#), [indices](#), and [type](#).

### 8.175.4.2 read\_archive()

```
void GiNaC::symmetry::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

Construct object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [indices](#), and [type](#).

#### 8.175.4.3 calchash()

```
unsigned GiNaC::symmetry::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [children](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [indices](#), [none](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [type](#).

#### 8.175.4.4 get\_type()

```
symmetry_type GiNaC::symmetry::get_type () const [inline]
```

Get symmetry type.

References [type](#).

#### 8.175.4.5 set\_type()

```
void GiNaC::symmetry::set_type (
    symmetry_type t) [inline]
```

Set symmetry type.

References [type](#).

Referenced by [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), and [GiNaC::sy\\_symm\(\)](#).

#### 8.175.4.6 add()

```
symmetry & GiNaC::symmetry::add (
    const symmetry & c)
```

Add child node, check index sets for consistency.

References [children](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [indices](#), [GiNaC::is\\_exactly\\_a\(\)](#), [none](#), [symmetry\(\)](#), and [type](#).

Referenced by [read\\_archive\(\)](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), [GiNaC::sy\\_cycl\(\)](#), [GiNaC::sy\\_none\(\)](#), [GiNaC::sy\\_none\(\)](#), [GiNaC::sy\\_symm\(\)](#), [GiNaC::sy\\_symm\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).



#### 8.175.4.7 validate()

```
void GiNaC::symmetry::validate (
    unsigned n)
```

Verify that all indices of this node are in the range [0..n-1].

This function throws an exception if the verification fails. If the top node has a type != none and no children, add all indices in the range [0..n-1] as children.

References [add\(\)](#), [indices](#), [none](#), and [type](#).

#### 8.175.4.8 has\_symmetry()

```
bool GiNaC::symmetry::has_symmetry () const [inline]
```

Check whether this node actually represents any kind of symmetry.

References [children](#), [none](#), and [type](#).

#### 8.175.4.9 has\_nonsymmetric()

```
bool GiNaC::symmetry::has_nonsymmetric () const
```

Check whether this node involves anything non symmetric.

References [antisymmetric](#), [children](#), [cyclic](#), [GiNaC::ex\\_to\(\)](#), [has\\_nonsymmetric\(\)](#), and [type](#).

Referenced by [has\\_nonsymmetric\(\)](#).

#### 8.175.4.10 has\_cyclic()

```
bool GiNaC::symmetry::has_cyclic () const
```

Check whether this node involves a cyclic symmetry.

References [children](#), [cyclic](#), [GiNaC::ex\\_to\(\)](#), [has\\_cyclic\(\)](#), and [type](#).

Referenced by [has\\_cyclic\(\)](#).

#### 8.175.4.11 do\_print()

```
void GiNaC::symmetry::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [antisymmetric](#), [children](#), [cyclic](#), [indices](#), [none](#), [GiNaC::print\\_context::s](#), [symmetric](#), and [type](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

#### 8.175.4.12 do\_print\_tree()

```
void GiNaC::symmetry::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [antisymmetric](#), [children](#), [cyclic](#), [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [indices](#), [none](#), [GiNaC::print\\_context::s](#), [symmetric](#), and [type](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.175.5 Friends And Related Symbol Documentation

#### 8.175.5.1 sy\_is\_less

```
friend class sy_is_less [friend]
```

References [sy\\_is\\_less](#).

Referenced by [canonicalize](#), and [sy\\_is\\_less](#).

#### 8.175.5.2 sy\_swap

```
friend class sy_swap [friend]
```

References [GiNaC::basic::basic\(\)](#), [sy\\_swap](#), [GiNaC::symm\(\)](#), and [symmetry\(\)](#).

Referenced by [canonicalize](#), and [sy\\_swap](#).

#### 8.175.5.3 canonicalize

```
int canonicalize (
    exvector::iterator v,
    const symmetry & symm) [friend]
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

##### Parameters

<i>v</i>	Start of expression vector
<a href="#">symm</a>	Root node of symmetry tree

##### Returns

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

References [antisymmetric](#), [GiNaC::canonicalize\(\)](#), [GiNaC::class\\_info< OPT >::first](#), [cyclic](#), [GiNaC::cyclic\\_permutation\(\)](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [GiNaC::shaker\\_sort\(\)](#), [sy\\_is\\_less](#), [sy\\_swap](#), [GiNaC::symm\(\)](#), [symmetric](#), and [symmetry\(\)](#).

## 8.175.6 Member Data Documentation

### 8.175.6.1 type

`symmetry_type` GiNaC::symmetry::type [private]

Type of symmetry described by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_type\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), [has\\_symmetry\(\)](#), [read\\_archive\(\)](#), [set\\_type\(\)](#), [symmetry\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).

### 8.175.6.2 indices

`std::set<unsigned>` GiNaC::symmetry::indices [private]

Sorted union set of all indices handled by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [read\\_archive\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).

### 8.175.6.3 children

`exvector` GiNaC::symmetry::children [private]

Vector of child nodes.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), and [has\\_symmetry\(\)](#).

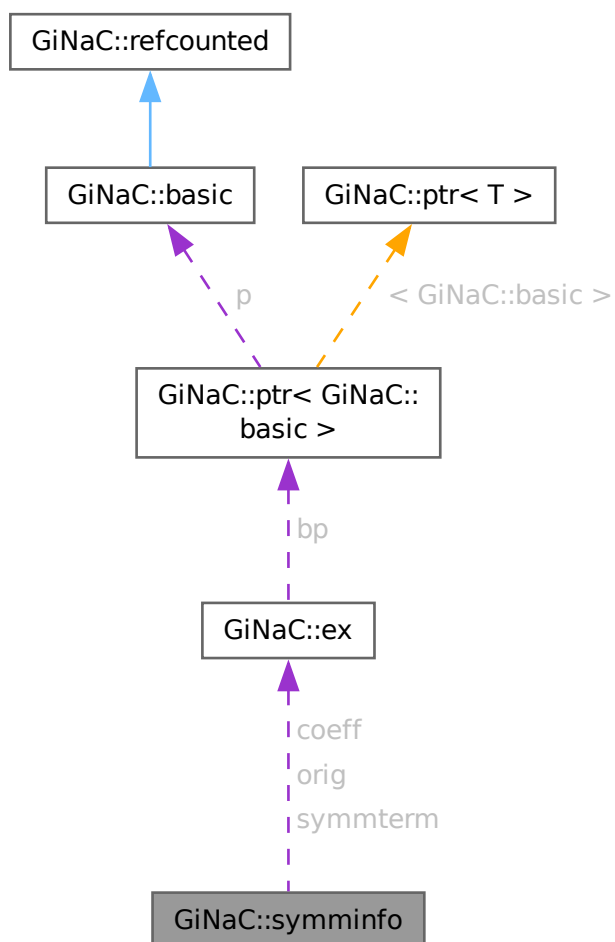
The documentation for this class was generated from the following files:

- [symmetry.h](#)
- [symmetry.cpp](#)

## 8.176 GiNaC::symminfo Class Reference

This structure stores the individual symmetrized terms obtained during the simplification of sums.

Collaboration diagram for `GiNaC::symminfo`:



### Public Member Functions

- `symminfo()`
- `symminfo(const ex &symmterm_, const ex &orig_, size_t num_)`

### Public Attributes

- `ex symmterm`  
*symmetrized term*
- `ex coeff`  
*coefficient of symmetrized term*
- `ex orig`  
*original term*
- `size_t num`  
*how many symmetrized terms resulted from the original term*

## 8.176.1 Detailed Description

This structure stores the individual symmetrized terms obtained during the simplification of sums.

## 8.176.2 Constructor & Destructor Documentation

### 8.176.2.1 symminfo() [1/2]

```
GiNaC::symminfo::symminfo () [inline]
```

References [num](#).

### 8.176.2.2 symminfo() [2/2]

```
GiNaC::symminfo::symminfo (  
    const ex & symmterm_,  
    const ex & orig_,  
    size_t num_) [inline]
```

References [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), [num](#), [GiNaC::ex::op\(\)](#), and [orig](#).

## 8.176.3 Member Data Documentation

### 8.176.3.1 symmterm

```
ex GiNaC::symminfo::symmterm
```

symmetrized term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)\(\)](#).

### 8.176.3.2 coeff

```
ex GiNaC::symminfo::coeff
```

coefficient of symmetrized term

### 8.176.3.3 orig

```
ex GiNaC::symminfo::orig
```

original term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)\(\)](#), and [symminfo\(\)](#).

### 8.176.3.4 num

```
size_t GiNaC::symminfo::num
```

how many symmetrized terms resulted from the original term

Referenced by [symminfo\(\)](#), and [symminfo\(\)](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 8.177 GiNaC::symminfo\_is\_less\_by\_orig Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

### 8.177.1 Member Function Documentation

#### 8.177.1.1 operator>()

```
bool GiNaC::symminfo_is_less_by_orig::operator() (
    const symminfo & si1,
    const symminfo & si2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::orig](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 8.178 GiNaC::symminfo\_is\_less\_by\_symmterm Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

### 8.178.1 Member Function Documentation

#### 8.178.1.1 operator>()

```
bool GiNaC::symminfo_is_less_by_symmterm::operator() (
    const symminfo & si1,
    const symminfo & si2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::symmterm](#).

The documentation for this class was generated from the following file:

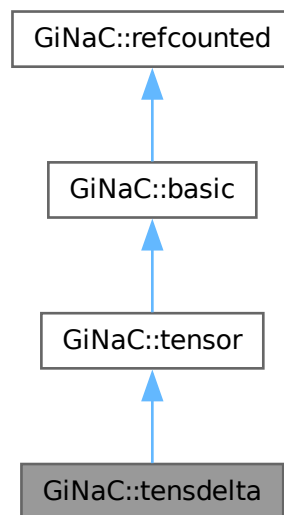
- [indexed.cpp](#)

## 8.179 GiNaC::tensdelta Class Reference

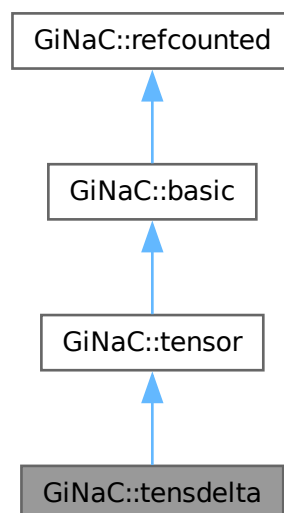
This class represents the delta tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensdelta:



Collaboration diagram for GiNaC::tensdelta:



## Public Member Functions

- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed delta tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed delta tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)



- virtual bool [has](#) (const [ex](#) &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned options=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int order, unsigned options=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- `const basic & hold` () const
- *Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const
- *Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `unsigned return_type` () const override
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- `unsigned return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual bool match_same_type` (const `basic` &other) const
- *Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &s) const
- *Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &other) const
- *Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const
- *Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const
- *Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.179.1 Detailed Description

This class represents the delta tensor.

If indexed, it must have exactly two indices of the same type.

## 8.179.2 Member Function Documentation

### 8.179.2.1 [info\(\)](#)

```
bool GiNaC::tensdelta::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [info\(\)](#), and [GiNaC::info\\_flags::real](#).

Referenced by [info\(\)](#).

### 8.179.2.2 eval\_indexed()

```
ex GiNaC::tensdelta::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed delta tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [eval\\_indexed\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [eval\\_indexed\(\)](#).

### 8.179.2.3 contract\_with()

```
bool GiNaC::tensdelta::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed delta tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 8.179.2.4 return\_type()

```
unsigned GiNaC::tensdelta::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.179.2.5 do\_print()

```
void GiNaC::tensdelta::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

### 8.179.2.6 do\_print\_latex()

```
void GiNaC::tensdelta::do_print_latex (
    const print_latex & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

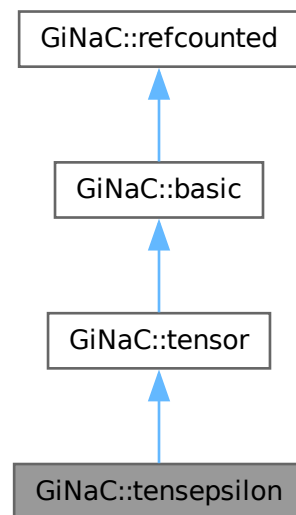
- [tensor.h](#)
- [tensor.cpp](#)

## 8.180 GiNaC::tensepsilon Class Reference

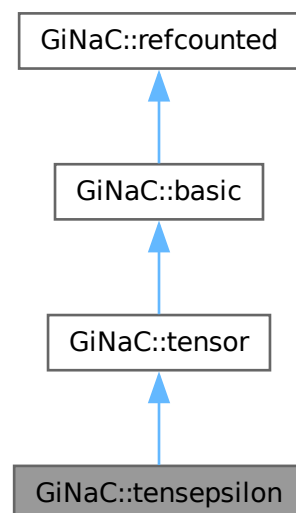
This class represents the totally antisymmetric epsilon tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensepsilon:



Collaboration diagram for GiNaC::tensepsilon:



## Public Member Functions

- `tensepsilon` (bool `minkowski`, bool `pos_sig`)
- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed epsilon tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of epsilon tensor with something else.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const

- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual `return_type_t return_type_info` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*

- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex_subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex_diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex_eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex_derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const



Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

- void [ensure\\_if\\_modifiable](#) () const

Ensure the object may be modified without hurting others, throws if this is not the case.

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

Default output to stream.

- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

Tree output to stream.

- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

Python parsable output to stream.

## Private Attributes

- bool [minkowski](#)

If true, tensor is in Minkowski-type space.

- bool [pos\\_sig](#)

If true, the metric is assumed to be  $\text{diag}(-1, 1, 1 \dots)$ .

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)

of type [status\\_flags](#)

- unsigned [hashvalue](#)

hash value

### 8.180.1 Detailed Description

This class represents the totally antisymmetric epsilon tensor.

If indexed, all indices must be of the same type and their number must be equal to the dimension of the index space.

### 8.180.2 Constructor & Destructor Documentation

#### 8.180.2.1 [tensepsilon\(\)](#)

```
GiNaC::tensepsilon::tensepsilon (
    bool minkowski,
    bool pos_sig)
```

References [minkowski](#), and [pos\\_sig](#).

### 8.180.3 Member Function Documentation

#### 8.180.3.1 info()

```
bool GiNaC::tensepsilon::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

#### 8.180.3.2 eval\_indexed()

```
ex GiNaC::tensepsilon::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed epsilon tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex\\_to\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_zero\(\)](#), [minkowski](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::permutation\\_sign\(\)](#), and [pos\\_sig](#).

#### 8.180.3.3 contract\_with()

```
bool GiNaC::tensepsilon::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of epsilon tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::lorentz\\_g\(\)](#), [GiNaC::metric\\_tensor\(\)](#), [minkowski](#), [pos\\_sig](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

#### 8.180.3.4 archive()

```
void GiNaC::tensepsilon::archive (
    archive\_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_bool\(\)](#), [minkowski](#), and [pos\\_sig](#).

### 8.180.3.5 read\_archive()

```
void GiNaC::tensepsilon::read_archive (
    const archive\_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::find\\_bool\(\)](#), [minkowski](#), and [pos\\_sig](#).

### 8.180.3.6 return\_type()

```
unsigned GiNaC::tensepsilon::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.180.3.7 do\_print()

```
void GiNaC::tensepsilon::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

### 8.180.3.8 do\_print\_latex()

```
void GiNaC::tensepsilon::do_print_latex (
    const print\_latex & c,
    unsigned level) const [protected]
```

## 8.180.4 Member Data Documentation

### 8.180.4.1 minkowski

```
bool GiNaC::tensepsilon::minkowski [private]
```

If true, tensor is in Minkowski-type space.

Otherwise it is in a Euclidean space.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), [read\\_archive\(\)](#), and [tensepsilon\(\)](#).

#### 8.180.4.2 pos\_sig

```
bool GiNaC::tensepsilon::pos_sig [private]
```

If true, the metric is assumed to be  $\text{diag}(-1, 1, 1, \dots)$ .

Otherwise it is  $\text{diag}(1, -1, -1, \dots)$ . This is only relevant if `minkowski = true`.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), [read\\_archive\(\)](#), and [tensepsilon\(\)](#).

The documentation for this class was generated from the following files:

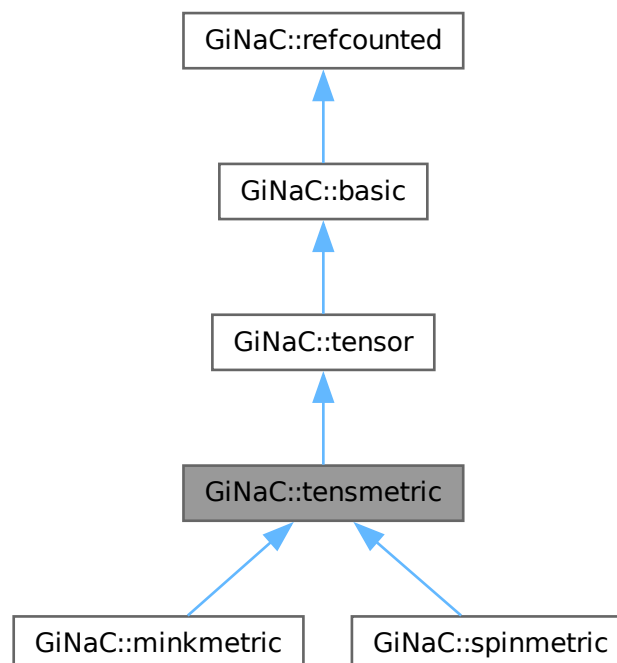
- [tensor.h](#)
- [tensor.cpp](#)

### 8.181 GiNaC::tensmetric Class Reference

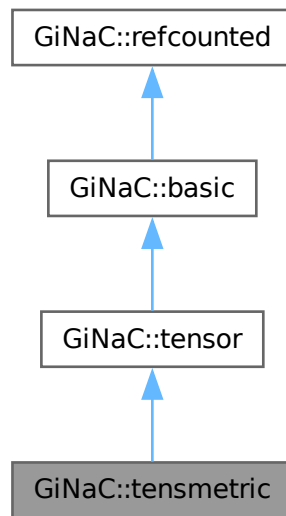
This class represents a general metric tensor which can be used to raise/lower indices.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensmetric:



Collaboration diagram for GiNaC::tensmetric:



### Public Member Functions

- bool `info` (unsigned inf) const override  
*Information about the object.*
- ex `eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual ex `eval` () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree ()` const  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 8.181.1 Detailed Description

This class represents a general metric tensor which can be used to raise/lower indices.

If indexed, it must have exactly two indices of the same type which must be of class `varidx` or a subclass.



## 8.181.2 Member Function Documentation

### 8.181.2.1 info()

```
bool GiNaC::tensmetric::info (
    unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 8.181.2.2 eval\_indexed()

```
ex GiNaC::tensmetric::eval_indexed (
    const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::delta\\_tensor\(\)](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::varidx::is\\_covariant\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::basic::subs\(\)](#).

### 8.181.2.3 contract\_with()

```
bool GiNaC::tensmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v) const [override], [virtual]
```

Contraction of an indexed metric tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 8.181.2.4 return\_type()

```
unsigned GiNaC::tensmetric::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 8.181.2.5 do\_print()

```
void GiNaC::tensmetric::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

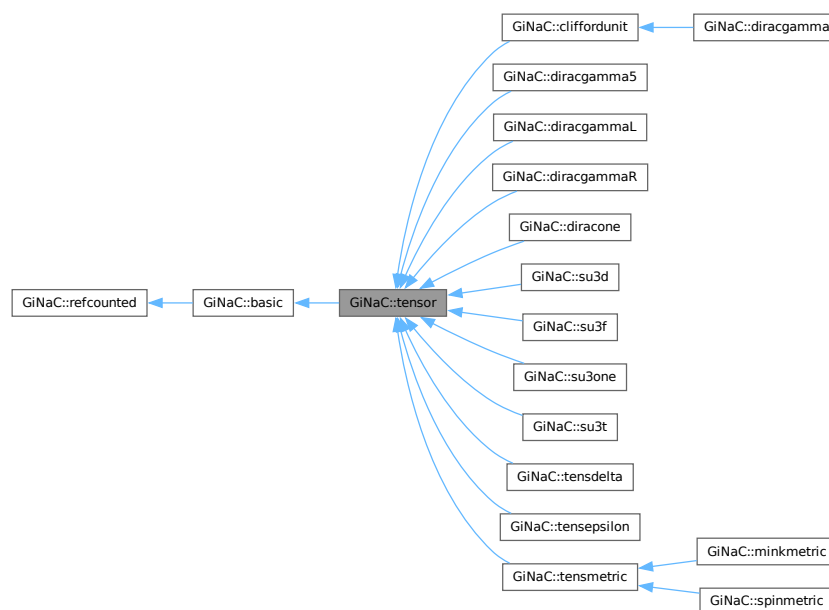
- [tensor.h](#)
- [tensor.cpp](#)

## 8.182 GiNaC::tensor Class Reference

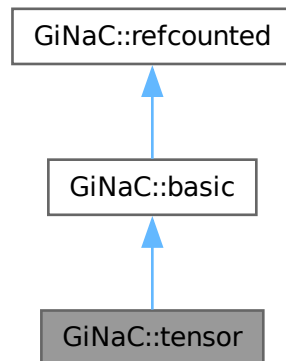
This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensor:



Collaboration diagram for GiNaC::tensor:



### Public Member Functions

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const basic &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const

- Little wrapper around printtree to be called within a debugger.*

  - virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual size\_t `nops` () const

*Number of operands/members.*
- virtual `ex op` (size\_t i) const

*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const

*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- *Try to contract two indexed expressions that appear in the same product.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T>
- void `print_dispatch` (const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const
- *Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- unsigned `return_type` () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.182.1 Detailed Description

This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

They are represented without indices. To attach indices to them, wrap them in an object of class [indexed](#).

### 8.182.2 Member Function Documentation

#### 8.182.2.1 [return\\_type\(\)](#)

```
unsigned GiNaC::tensor::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

## 8.182.2.2 replace\_contr\_index()

```
bool GiNaC::tensor::replace_contr_index (
    exvector::iterator self,
    exvector::iterator other) const
```

Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).

References [GiNaC::\\_ex1](#), [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), [GiNaC::is\\_a\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::tensdelta::contract\\_with\(\)](#), and [GiNaC::tensmetric::contract\\_with\(\)](#).

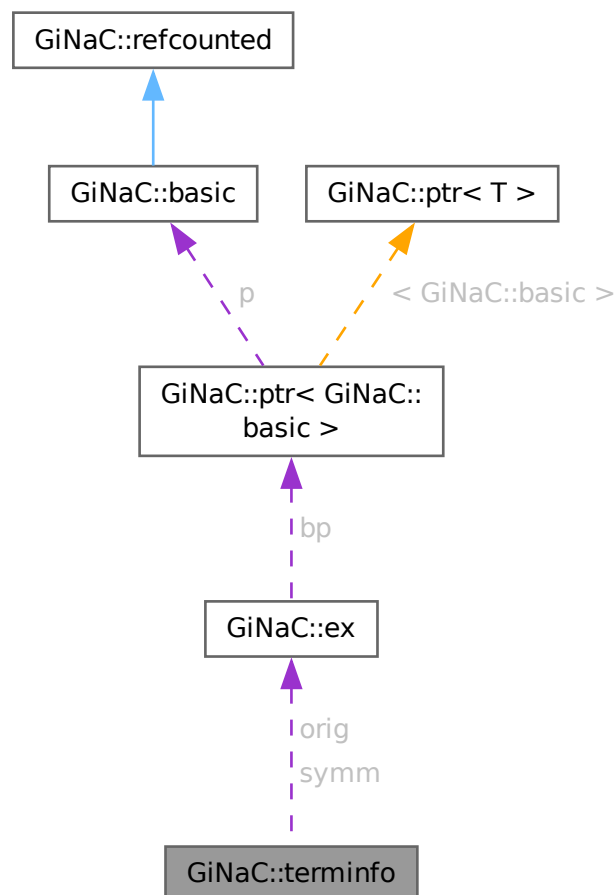
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 8.183 GiNaC::terminfo Class Reference

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

Collaboration diagram for GiNaC::terminfo:



## Public Member Functions

- [terminfo](#) (const [ex](#) &orig\_, const [ex](#) &symm\_)

## Public Attributes

- [ex orig](#)  
*original term*
- [ex symm](#)  
*symmetrized term*

### 8.183.1 Detailed Description

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

### 8.183.2 Constructor & Destructor Documentation

#### 8.183.2.1 [terminfo\(\)](#)

```
GiNaC::terminfo::terminfo (  
    const ex & orig_,  
    const ex & symm_) [inline]
```

References [orig](#), and [symm](#).

### 8.183.3 Member Data Documentation

#### 8.183.3.1 [orig](#)

```
ex GiNaC::terminfo::orig
```

original term

Referenced by [terminfo\(\)](#).

#### 8.183.3.2 [symm](#)

```
ex GiNaC::terminfo::symm
```

symmetrized term

Referenced by [GiNaC::terminfo\\_is\\_less::operator>\(\)](#), and [terminfo\(\)](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)



## 8.184 GiNaC::terminfo\_is\_less Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [terminfo](#) &ti1, const [terminfo](#) &ti2) const

### 8.184.1 Member Function Documentation

#### 8.184.1.1 operator>()()

```
bool GiNaC::terminfo_is_less::operator() (
    const terminfo & ti1,
    const terminfo & ti2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::terminfo::symm](#).

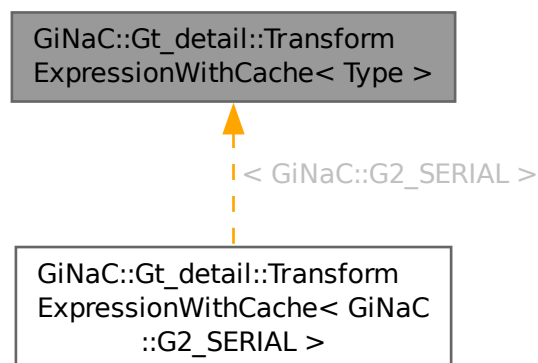
The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 8.185 GiNaC::Gt\_detail::TransformExpressionWithCache< Type > Class Template Reference

```
#include <Gt_helpers.h>
```

Inheritance diagram for GiNaC::Gt\_detail::TransformExpressionWithCache< Type >:



## Public Member Functions

- [TransformExpressionWithCache](#) (const std::function< [ex](#)(const [ex](#) &)> &[func\\_obj](#), const std::function< [ex](#)(const [ex](#) &)> &[func\\_default](#)=[ ](const [ex](#) &obj) -> [ex](#) { return obj;})
- [ex operator\(\)](#) (const [ex](#) &input)
- void [print\\_cache](#) () const
- const [exhashmap](#)< [ex](#) > & [get\\_cache](#) () const
- void [clear\\_cache](#) ()

## Private Member Functions

- [ex impl](#) (const [ex](#) &expr)

## Static Private Member Functions

- template<typename T = Type>  
static std::enable\_if< std::is\_base\_of< [basic](#), T >::value, bool >::type [check\\_type](#) (const [ex](#) &expr)
- template<typename T = Type>  
static std::enable\_if<!std::is\_base\_of< [basic](#), T >::value, bool >::type [check\\_type](#) (const [ex](#) &expr)

## Private Attributes

- const std::function< [ex](#)(const [ex](#) &)> [func\\_obj](#)
- const std::function< [ex](#)(const [ex](#) &)> [func\\_default](#)
- [exhashmap](#)< [ex](#) > [cache](#)

## 8.185.1 Constructor & Destructor Documentation

### 8.185.1.1 TransformExpressionWithCache()

```
template<typename Type>
GiNaC::Gt_detail::TransformExpressionWithCache< Type >::TransformExpressionWithCache (
    const std::function< ex(const ex &)> & func\_obj,
    const std::function< ex(const ex &)> & func\_default = [] (const ex& obj) -> ex { return obj; })
[inline]
```

References [func\\_default](#), and [func\\_obj](#).

## 8.185.2 Member Function Documentation

### 8.185.2.1 operator>()

```
template<typename Type>
ex GiNaC::Gt_detail::TransformExpressionWithCache< Type >::operator() (
    const ex & input) [inline]
```

References [impl](#)() .

### 8.185.2.2 print\_cache()

```
template<typename Type>
void GiNaC::Gt_detail::TransformExpressionWithCache< Type >::print_cache () const
```

References [cache](#).

### 8.185.2.3 get\_cache()

```
template<typename Type>
const exhashmap< ex > & GiNaC::Gt_detail::TransformExpressionWithCache< Type >::get_cache ()
const [inline]
```

References [cache](#).

### 8.185.2.4 clear\_cache()

```
template<typename Type>
void GiNaC::Gt_detail::TransformExpressionWithCache< Type >::clear_cache () [inline]
```

References [cache](#).

### 8.185.2.5 check\_type() [1/2]

```
template<typename Type>
template<typename T>
std::enable_if<!std::is_base_of< basic, T >::value, bool >::type GiNaC::Gt_detail::TransformExpressionWithCa
Type >::check_type (
    const ex & expr) [static], [private]
```

References [GiNaC::is\\_exactly\\_a\(\)](#).

Referenced by [impl\(\)](#).

### 8.185.2.6 check\_type() [2/2]

```
template<typename Type>
template<typename T = Type>
std::enable_if<!std::is_base_of< basic, T >::value, bool >::type GiNaC::Gt_detail::TransformExpressionWithCa
Type >::check_type (
    const ex & expr) [static], [private]
```

### 8.185.2.7 impl()

```
template<typename Type>
ex GiNaC::Gt_detail::TransformExpressionWithCache< Type >::impl (
    const ex & expr) [private]
```

References [GiNaC::container< C >::append\(\)](#), [cache](#), [check\\_type\(\)](#), [func\\_default](#), [func\\_obj](#), [impl\(\)](#), [GiNaC::is\\_exactly\\_a\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [impl\(\)](#), and [operator\(\)\(\)](#).

### 8.185.3 Member Data Documentation

#### 8.185.3.1 func\_obj

```
template<typename Type>
const std::function<ex(const ex&)> GiNaC::Gt_detail::TransformExpressionWithCache< Type >::↔
func_obj [private]
```

Referenced by [impl\(\)](#), and [TransformExpressionWithCache\(\)](#).

#### 8.185.3.2 func\_default

```
template<typename Type>
const std::function<ex(const ex&)> GiNaC::Gt_detail::TransformExpressionWithCache< Type >::↔
func_default [private]
```

Referenced by [impl\(\)](#), and [TransformExpressionWithCache\(\)](#).

#### 8.185.3.3 cache

```
template<typename Type>
exhashmap<ex> GiNaC::Gt_detail::TransformExpressionWithCache< Type >::cache [private]
```

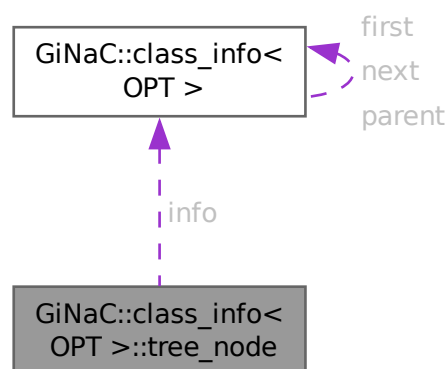
Referenced by [clear\\_cache\(\)](#), [get\\_cache\(\)](#), [impl\(\)](#), and [print\\_cache\(\)](#).

The documentation for this class was generated from the following files:

- [Gt.h](#)
- [Gt\\_helpers.h](#)
- [Gt\\_helpers.cpp](#)

## 8.186 GiNaC::class\_info< OPT >::tree\_node Struct Reference

Collaboration diagram for GiNaC::class\_info< OPT >::tree\_node:



## Public Member Functions

- [tree\\_node](#) ([class\\_info](#) \*i)
- void [add\\_child](#) ([tree\\_node](#) \*n)

## Public Attributes

- `std::vector< tree\_node * >` [children](#)
- [class\\_info](#) \* [info](#)

## 8.186.1 Constructor & Destructor Documentation

### 8.186.1.1 tree\_node()

```
template<class OPT>
GiNaC::class_info< OPT >::tree_node::tree_node (
    class_info * i) [inline]
```

References [GiNaC::class\\_info< OPT >::class\\_info\(\)](#), and [info](#).

Referenced by [add\\_child\(\)](#).

## 8.186.2 Member Function Documentation

### 8.186.2.1 add\_child()

```
template<class OPT>
void GiNaC::class_info< OPT >::tree_node::add_child (
    tree_node * n) [inline]
```

References [children](#), and [tree\\_node\(\)](#).

## 8.186.3 Member Data Documentation

### 8.186.3.1 children

```
template<class OPT>
std::vector<tree\_node *> GiNaC::class_info< OPT >::tree_node::children
```

Referenced by [add\\_child\(\)](#), and [GiNaC::class\\_info< OPT >::dump\\_tree\(\)](#).

### 8.186.3.2 info

```
template<class OPT>
class_info* GiNaC::class_info< OPT >::tree_node::info
```

Referenced by [GiNaC::class\\_info< OPT >::dump\\_tree\(\)](#), and [tree\\_node\(\)](#).

The documentation for this struct was generated from the following file:

- [class\\_info.h](#)

## 8.187 GiNaC::unarchive\_table\_t Class Reference

```
#include <archive.h>
```

### Public Member Functions

- [unarchive\\_table\\_t\(\)](#)
- [~unarchive\\_table\\_t\(\)](#)
- [synthesize\\_func find](#) (const std::string &classname) const
- void [insert](#) (const std::string &classname, [synthesize\\_func](#) f)

### Static Private Attributes

- static int [usecount](#) = 0
- static [unarchive\\_map\\_t](#) \* [unarch\\_map](#) = nullptr

### 8.187.1 Constructor & Destructor Documentation

#### 8.187.1.1 unarchive\_table\_t()

```
GiNaC::unarchive_table_t::unarchive_table_t ()
```

References [unarch\\_map](#), and [usecount](#).

#### 8.187.1.2 ~unarchive\_table\_t()

```
GiNaC::unarchive_table_t::~~unarchive_table_t ()
```

References [unarch\\_map](#), and [usecount](#).

### 8.187.2 Member Function Documentation

#### 8.187.2.1 find()

```
synthesize\_func GiNaC::unarchive_table_t::find (  
    const std::string & classname) const
```

References [unarch\\_map](#).

Referenced by [GiNaC::find\\_factory\\_fcn\(\)](#).

#### 8.187.2.2 insert()

```
void GiNaC::unarchive_table_t::insert (  
    const std::string & classname,  
    synthesize\_func f)
```

References [unarch\\_map](#).

### 8.187.3 Member Data Documentation

#### 8.187.3.1 usecount

```
int GiNaC::unarchive_table_t::usecount = 0 [static], [private]
```

Referenced by [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

#### 8.187.3.2 unarch\_map

```
unarchive_map_t * GiNaC::unarchive_table_t::unarch_map = nullptr [static], [private]
```

Referenced by [find\(\)](#), [insert\(\)](#), [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

The documentation for this class was generated from the following files:

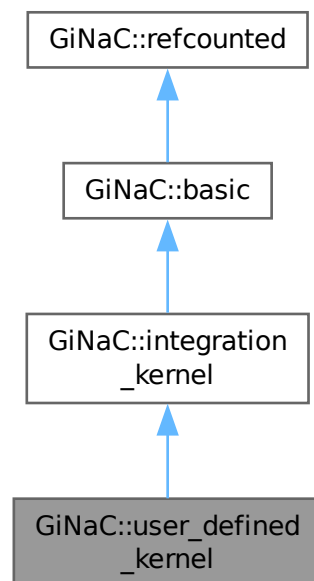
- [archive.h](#)
- [archive.cpp](#)

## 8.188 GiNaC::user\_defined\_kernel Class Reference

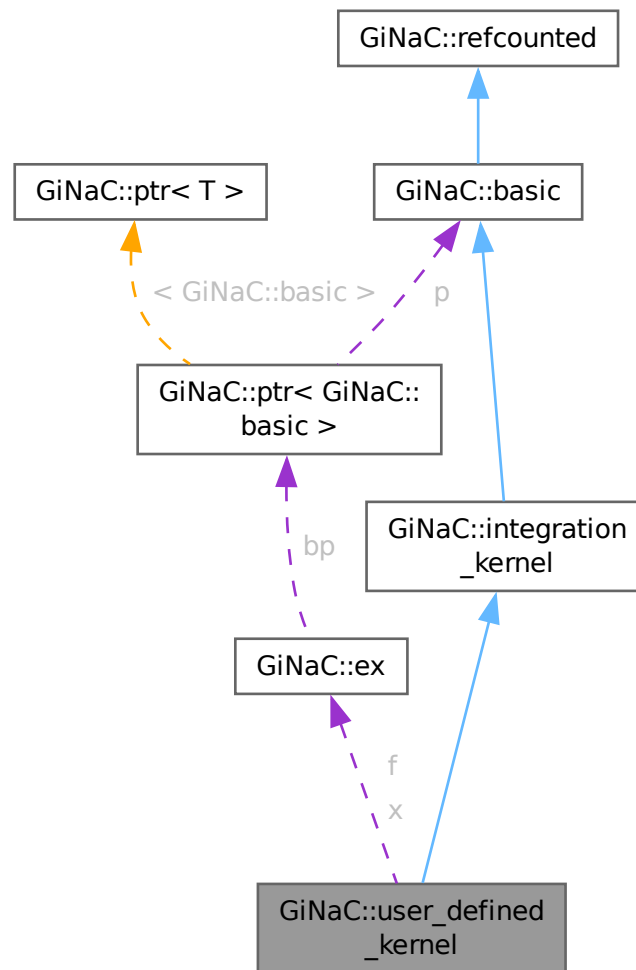
A user-defined integration kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::user\_defined\_kernel:



Collaboration diagram for `GiNaC::user_defined_kernel`:



## Public Member Functions

- `user_defined_kernel` (const `ex` &`f`, const `ex` &`x`)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `bool is_numeric (void)` const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`x`, int order) const override  
*Returns the Laurent series, starting possibly with the pole term.*



## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const **relational** &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex get\_numerical\_value** (const **ex** &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const **basic** &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const **basic** &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const **print\_context** &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const **ex** &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const **ex** &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const **ex** &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned options=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned options=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T>

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Load (deserialize) the object from an archive node.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Helper function for `subs()`.*
- `int compare` (const `basic` &other) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `bool is_equal` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `bool uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex f](#)
- [ex x](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.188.1 Detailed Description

A user-defined integration kernel.

The input is an expression  $f$ , depending on a variable  $x$ . It is assumed that  $f$  has a Laurent expansion around  $x = 0$  and maximally a simple pole at  $x = 0$ .

## 8.188.2 Constructor & Destructor Documentation

### 8.188.2.1 user\_defined\_kernel()

```
GiNaC::user_defined_kernel::user_defined_kernel (
    const ex & f,
    const ex & x)
```

References [f](#), [user\\_defined\\_kernel\(\)](#), and [x](#).

Referenced by [user\\_defined\\_kernel\(\)](#).

## 8.188.3 Member Function Documentation

### 8.188.3.1 nops()

```
size_t GiNaC::user_defined_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 8.188.3.2 op()

```
ex GiNaC::user_defined_kernel::op (
    size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [f](#), and [x](#).

### 8.188.3.3 let\_op()

```
ex & GiNaC::user_defined_kernel::let_op (
    size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [f](#), and [x](#).

### 8.188.3.4 is\_numeric()

```
bool GiNaC::user_defined_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [f](#), [GiNaC::info\\_flags::numeric](#), and [x](#).

### 8.188.3.5 Laurent\_series()

```
ex GiNaC::user_defined_kernel::Laurent_series (
    const ex & x,
    int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [f](#), and [x](#).

### 8.188.3.6 uses\_Laurent\_series()

```
bool GiNaC::user_defined_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 8.188.3.7 do\_print()

```
void GiNaC::user_defined_kernel::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [f](#), [GiNaC::print\\_context::s](#), and [x](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.188.4 Member Data Documentation

### 8.188.4.1 f

```
ex GiNaC::user_defined_kernel::f [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [user\\_defined\\_kernel\(\)](#).

### 8.188.4.2 x

```
ex GiNaC::user_defined_kernel::x [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [user\\_defined\\_kernel\(\)](#).

The documentation for this class was generated from the following files:

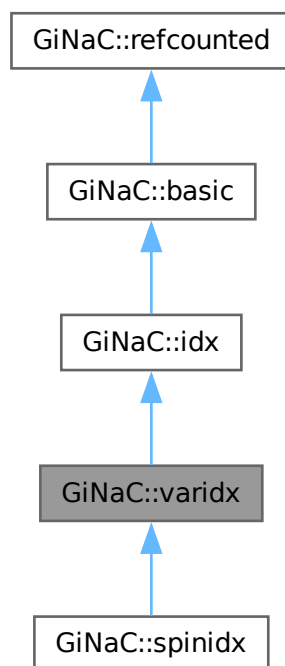
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 8.189 GiNaC::varidx Class Reference

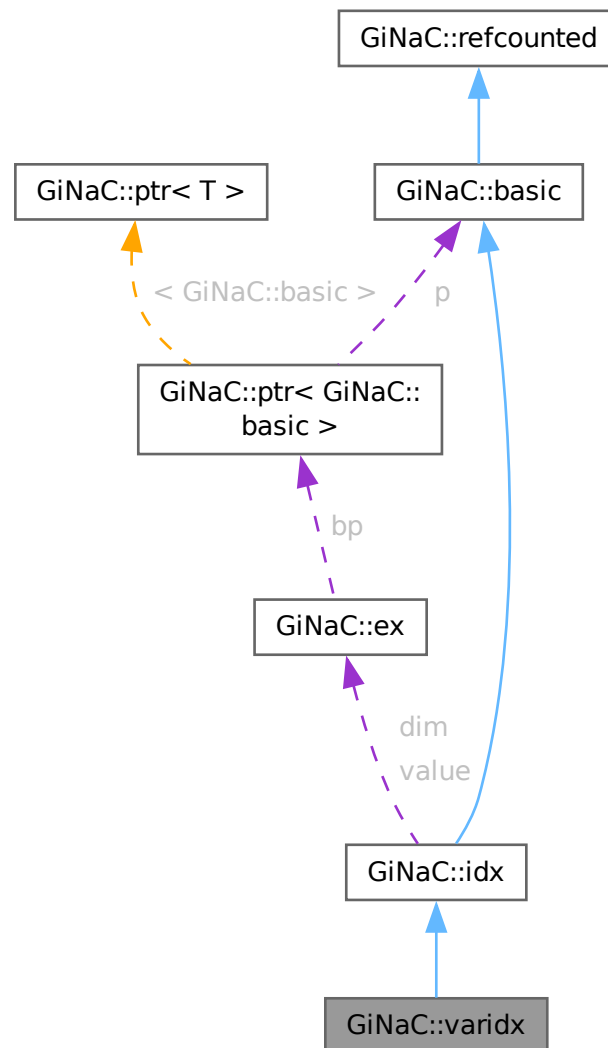
This class holds an index with a variance (co- or contravariant).

```
#include <idx.h>
```

Inheritance diagram for GiNaC::varidx:



Collaboration diagram for `GiNaC::varidx`:



## Public Member Functions

- `varidx` (const `ex` &`v`, const `ex` &`dim`, bool `covariant`=false)  
Construct index with given value, dimension and variance.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
Load (deserialize) the object from an archive node.
- bool `is_covariant` () const  
Check whether the index is covariant.



- `bool is_contravariant () const`  
*Check whether the index is contravariant (not covariant).*
- `ex toggle_variance () const`  
*Make a new index with the same value but the opposite variance.*

## Public Member Functions inherited from `GiNaC::idx`

- `idx (const ex &v, const ex &dim)`  
*Construct index with given value and dimension.*
- `bool info (unsigned inf) const override`  
*Information about the object.*
- `size_t nops () const override`  
*Number of operands/members.*
- `ex op (size_t i) const override`  
*Return operand/member at position i.*
- `ex map (map_function &f) const override`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf () const override`  
*By default, `basic::evalf` would evaluate the index value but we don't want a.1 to become a.*
- `ex subs (const exmap &m, unsigned options=0) const override`  
*Substitute a set of objects by arbitrary expressions.*
- `ex get_value () const`  
*Get value of index.*
- `bool is_numeric () const`  
*Check whether the index is numeric.*
- `bool is_symbolic () const`  
*Check whether the index is symbolic.*
- `ex get_dim () const`  
*Get dimension of index space.*
- `bool is_dim_numeric () const`  
*Check whether the dimension is numeric.*
- `bool is_dim_symbolic () const`  
*Check whether the dimension is symbolic.*
- `ex replace_dim (const ex &new_dim) const`  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim (const idx &other) const`  
*Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree ()` const  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*

- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T>  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned options) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::idx](#)

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an index always returns 0.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_index](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- bool [covariant](#)  
 *$x.mu$ , default is contravariant:  $x \sim mu$*

## Protected Attributes inherited from [GiNaC::idx](#)

- [ex value](#)  
*Expression that constitutes the index (numeric or symbolic name).*
- [ex dim](#)  
*Dimension of space (can be symbolic or numeric).*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.189.1 Detailed Description

This class holds an index with a variance (co- or contravariant).

There is an associated metric tensor that can be used to raise/lower indices.

### 8.189.2 Constructor & Destructor Documentation

#### 8.189.2.1 varidx()

```
GiNaC::varidx::varidx (
    const ex & v,
    const ex & dim,
    bool covariant = false)
```

Construct index with given value, dimension and variance.

#### Parameters

<a href="#">v</a>	Value of index (numeric or symbolic)
<a href="#">dim</a>	Dimension of index space (numeric or symbolic)
<a href="#">covariant</a>	Make covariant index (default is contravariant)

References [covariant](#).

Referenced by [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), and [toggle\\_variance\(\)](#).

### 8.189.3 Member Function Documentation

#### 8.189.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::varidx::is_dummy_pair_same_type (
    const basic & other) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

References [covariant](#), and [varidx\(\)](#).

#### 8.189.3.2 archive()

```
void GiNaC::varidx::archive (
    archive\_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

References [GiNaC::archive\\_node::add\\_bool\(\)](#), and [covariant](#).

### 8.189.3.3 read\_archive()

```
void GiNaC::varidx::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

References [covariant](#), and [GiNaC::archive\\_node::find\\_bool\(\)](#).

### 8.189.3.4 match\_same\_type()

```
bool GiNaC::varidx::match_same_type (
    const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

References [covariant](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_a\(\)](#), and [varidx\(\)](#).

### 8.189.3.5 is\_covariant()

```
bool GiNaC::varidx::is_covariant () const [inline]
```

Check whether the index is covariant.

References [covariant](#).

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

### 8.189.3.6 is\_contravariant()

```
bool GiNaC::varidx::is_contravariant () const [inline]
```

Check whether the index is contravariant (not covariant).

References [covariant](#).

### 8.189.3.7 toggle\_variance()

```
ex GiNaC::varidx::toggle_variance () const
```

Make a new index with the same value but the opposite variance.

References [GiNaC::basic::clearflag\(\)](#), [covariant](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), and [varidx\(\)](#).

Referenced by [GiNaC::base\\_and\\_index\(\)](#), and [GiNaC::clifford::eval\\_ncmul\(\)](#).

### 8.189.3.8 do\_print()

```
void GiNaC::varidx::do_print (
    const print_context & c,
    unsigned level) const [protected]
```

References [covariant](#), [GiNaC::idx::print\\_index\(\)](#), and [GiNaC::print\\_context::s](#).

### 8.189.3.9 do\_print\_tree()

```
void GiNaC::varidx::do_print_tree (
    const print_tree & c,
    unsigned level) const [protected]
```

References [covariant](#), [GiNaC::print\\_tree::delta\\_indent](#), [GiNaC::idx::dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::print\\_context::s](#), and [GiNaC::idx::value](#).

## 8.189.4 Member Data Documentation

### 8.189.4.1 covariant

```
bool GiNaC::varidx::covariant [protected]
```

x.mu, default is contravariant:  $x \sim \mu$

Referenced by [archive\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), [is\\_contravariant\(\)](#), [is\\_covariant\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_variance\(\)](#), [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#), and [varidx\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

## 8.190 GiNaC::visitor Class Reference

Degenerate base class for visitors.

```
#include <basic.h>
```

### Protected Member Functions

- virtual [~visitor](#) ()

### 8.190.1 Detailed Description

Degenerate base class for visitors.

basic and derivative classes support Robert C. Martin's Acyclic Visitor pattern (cf. <http://condor.depaul.edu/dmumaugh/OOT/Design-Principles/acv.pdf> or chapter 10 of Andrei Alexandrescu's "Modern C++ Design").

### 8.190.2 Constructor & Destructor Documentation

#### 8.190.2.1 ~visitor()

```
virtual GiNaC::visitor::~~visitor () [inline], [protected], [virtual]
```

The documentation for this class was generated from the following file:

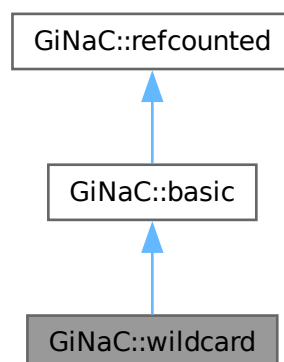
- [basic.h](#)

## 8.191 GiNaC::wildcard Class Reference

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

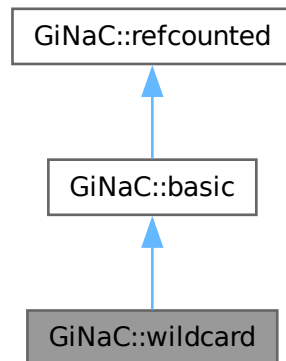
```
#include <wildcard.h>
```

Inheritance diagram for GiNaC::wildcard:





Collaboration diagram for GiNaC::wildcard:



### Public Member Functions

- `wildcard` (unsigned `label`)  
*Construct wildcard with specified label.*
- `bool match` (const `ex` &pattern, `exmap` &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- `void archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- `void read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- unsigned `get_label` () const

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const

- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

  - virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const

*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual size\_t `nops` () const

*Number of operands/members.*
- virtual `ex op` (size\_t i) const

*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned options=0) const

*Test for occurrence of a pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned options=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned options=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int order, unsigned options=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

- Implementation *ex::max\_coefficient()*.
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type ()` const
- virtual `return_type_t return_type_info ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T>  
`void print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other)` const  
*Test for syntactic equality.*
- `const basic & hold ()` const  
*Stop further evaluation.*
- `unsigned gethash ()` const
- `const basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- `refcounted ()` noexcept
- `unsigned int add_reference ()` noexcept
- `unsigned int remove_reference ()` noexcept
- `unsigned int get_refcount ()` const noexcept
- `void set_refcount (unsigned int r)` noexcept

## Protected Member Functions

- `unsigned calchash ()` const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void do_print (const print_context &c, unsigned level)` const
- `void do_print_tree (const print_tree &c, unsigned level)` const
- `void do_print_python_repr (const print_python_repr &c, unsigned level)` const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- unsigned [label](#)  
*Label used to distinguish different wildcards.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 8.191.1 Detailed Description

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

An integer label is used to identify different wildcards.

### 8.191.2 Constructor & Destructor Documentation

#### 8.191.2.1 wildcard()

```
GiNaC::wildcard::wildcard (
    unsigned label)
```

Construct wildcard with specified label.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [label](#), [GiNaC::basic::setflag\(\)](#), and [wildcard\(\)](#).

Referenced by [wildcard\(\)](#).

## 8.191.3 Member Function Documentation

### 8.191.3.1 match()

```
bool GiNaC::wildcard::match (
    const ex & pattern,
    exmap & repl_lst) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#), [GiNaC::ex\\_to\(\)](#), and [GiNaC::basic::is\\_equal\(\)](#).

### 8.191.3.2 archive()

```
void GiNaC::wildcard::archive (
    archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::archive\\_node::add\\_unsigned\(\)](#), and [label](#).

### 8.191.3.3 read\_archive()

```
void GiNaC::wildcard::read_archive (
    const archive_node & n,
    lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), [label](#), and [GiNaC::basic::setflag\(\)](#).

### 8.191.3.4 calchash()

```
unsigned GiNaC::wildcard::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [label](#), and [GiNaC::basic::setflag\(\)](#).

### 8.191.3.5 `get_label()`

```
unsigned GiNaC::wildcard::get_label () const [inline]
```

References [label](#).

### 8.191.3.6 `do_print()`

```
void GiNaC::wildcard::do_print (
    const print\_context & c,
    unsigned level) const [protected]
```

References [label](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.191.3.7 `do_print_tree()`

```
void GiNaC::wildcard::do_print_tree (
    const print\_tree & c,
    unsigned level) const [protected]
```

References [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [label](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

### 8.191.3.8 `do_print_python_repr()`

```
void GiNaC::wildcard::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level) const [protected]
```

References [label](#), and [GiNaC::print\\_context::s](#).

Referenced by [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#).

## 8.191.4 Member Data Documentation

### 8.191.4.1 `label`

```
unsigned GiNaC::wildcard::label [private]
```

Label used to distinguish different wildcards.

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_label\(\)](#), [read\\_archive\(\)](#), and [wildcard\(\)](#).

The documentation for this class was generated from the following files:

- [wildcard.h](#)
- [wildcard.cpp](#)

## 8.192 GiNaC::zeta1\_SERIAL Class Reference

Complex conjugate.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 8.192.1 Detailed Description

Complex conjugate.

Real part. Imaginary part. Absolute value. Step function. Complex sign. Eta function:  $\log(a*b) == \log(a) + \log(b) + \eta(a, b)$ . Sine. Cosine. Tangent. Exponential function. Natural logarithm. Inverse sine (arc sine). Inverse cosine (arc cosine). Inverse tangent (arc tangent). Inverse tangent with two arguments. Hyperbolic Sine. Hyperbolic Cosine. Hyperbolic Tangent. Inverse hyperbolic Sine (area hyperbolic sine). Inverse hyperbolic Cosine (area hyperbolic cosine). Inverse hyperbolic Tangent (area hyperbolic tangent). Dilogarithm. Trilogarithm. Derivatives of Riemann's Zeta-function. Multiple zeta value including Riemann's zeta-function.

### 8.192.2 Member Data Documentation

#### 8.192.2.1 serial

```
unsigned GiNaC::zeta1_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("zeta", 1).
    evalf_func(zeta1_evalf).
    eval_func(zeta1_eval).
    derivative_func(zeta1_deriv).
    print_func<print_latex>(zeta1_print_latex).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 8.193 GiNaC::zeta2\_SERIAL Class Reference

Alternating Euler sum or colored MZV.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

## 8.193.1 Detailed Description

Alternating Euler sum or colored MZV.

## 8.193.2 Member Data Documentation

### 8.193.2.1 [serial](#)

unsigned `GiNaC::zeta2_SERIAL::serial` [static]

#### Initial value:

```
= function::register_new(function_options("zeta", 2).
    evalf_func(zeta2_evalf).
    eval_func(zeta2_eval).
    derivative_func(zeta2_deriv).
    print_func<print_latex>(zeta2_print_latex).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)



## Chapter 9

# File Documentation

### 9.1 add.cpp File Reference

Implementation of [GiNaC](#)'s sums of expressions.

```
#include "add.h"
#include "mul.h"
#include "archive.h"
#include "operators.h"
#include "matrix.h"
#include "utils.h"
#include "clifford.h"
#include "ncmul.h"
#include "compiler.h"
#include <limits>
#include <stdexcept>
```

#### Namespaces

- namespace [GiNaC](#)

#### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (add, expairseq, print\_func< [print\\_context](#) >(&add::do\_print). print\_func< [print\\_latex](#) >(&add::do\_print\_latex). print\_func< [print\\_csrc](#) >(&add::do\_print\_csrc). print\_func< [print\\_tree](#) >(&add::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&add::do\_print\_python\_repr))  
add
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (add)

#### 9.1.1 Detailed Description

Implementation of [GiNaC](#)'s sums of expressions.

## 9.2 add.h File Reference

Interface to [GiNaC](#)'s sums of expressions.

```
#include "expairseq.h"
```

### Classes

- class [GiNaC::add](#)  
*Sum of expressions.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([add](#))

### 9.2.1 Detailed Description

Interface to [GiNaC](#)'s sums of expressions.

## 9.3 archive.cpp File Reference

Archiving of [GiNaC](#) expressions.

```
#include "archive.h"  
#include "registrar.h"  
#include "ex.h"  
#include "lst.h"  
#include "version.h"  
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

## Functions

- static void [GiNaC::write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [GiNaC::read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [archive\\_node](#) &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [archive](#) &ar)  
*Write archive to binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, [archive\\_node](#) &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, [archive](#) &ar)  
*Read archive from binary data stream.*
- static [synthesize\\_func](#) [GiNaC::find\\_factory\\_fcn](#) (const std::string &name)

### 9.3.1 Detailed Description

Archiving of [GiNaC](#) expressions.

## 9.4 archive.h File Reference

Archiving of [GiNaC](#) expressions.

```
#include "ex.h"
#include <iosfwd>
#include <map>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).*
- struct [GiNaC::archive\\_node::property\\_info](#)  
*Information about a stored property.*
- struct [GiNaC::archive\\_node::property](#)  
*Archived property (data type, name and associated data).*
- struct [GiNaC::archive\\_node::archive\\_node\\_cit\\_range](#)
- class [GiNaC::unarchive\\_table\\_t](#)
- class [GiNaC::archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).*
- struct [GiNaC::archive::archived\\_ex](#)  
*Archived expression descriptor.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define `GINAC_DECLARE_UNARCHIVER`(classname)  
*Helper macros to register a class with (un)archiving (a.k.a.*
- #define `GINAC_BIND_UNARCHIVER`(classname)

## Typedefs

- typedef unsigned `GiNaC::archive_node_id`  
*Numerical ID value to refer to an [archive\\_node](#).*
- typedef unsigned `GiNaC::archive_atom`  
*Numerical ID value to refer to a string.*
- typedef `basic` \*(\* `GiNaC::synthesize_func`) ()
- typedef `std::map< std::string, synthesize_func >` `GiNaC::unarchive_map_t`

## Functions

- `std::ostream & GiNaC::operator<<` (`std::ostream &os`, const `archive` &ar)  
*Write archive to binary data stream.*
- `std::istream & GiNaC::operator>>` (`std::istream &is`, `archive` &ar)  
*Read archive from binary data stream.*

## Variables

- static `unarchive_table_t` `GiNaC::unarch_table_instance`

## 9.4.1 Detailed Description

Archiving of `GiNaC` expressions.

## 9.4.2 Macro Definition Documentation

### 9.4.2.1 GINAC\_DECLARE\_UNARCHIVER

```
#define GINAC_DECLARE_UNARCHIVER(  
    classname)
```

#### Value:

```
class classname ## _unarchiver \
{ \
    static int usecount; \
public: \
    static GiNaC::basic* create(); \
    classname ## _unarchiver(); \
    ~ classname ## _unarchiver(); \
}; \
static classname ## _unarchiver classname ## _unarchiver_instance
```

Helper macros to register a class with (un)archiving (a.k.a.

(de)serialization).

Usage: put

`GINAC_DECLARE_UNARCHIVER(myclass);`

into the header file (in the global or namespace scope), and

`GINAC_BIND_UNARCHIVER(myclass);`

into the source file.

Effect: the `'myclass'` (being a class derived directly or indirectly from `GiNaC::basic`) can be archived and unarchived.

Note: you need to use `GINAC_{DECLARE,BIND}_UNARCHIVER` incantations in order to make your class (un)archivable *even if your class does not overload `'read_archive'` method*. Sorry for inconvenience.

How it works:

The `basic` class has a `read_archive` virtual method which reads an expression from archive. Derived classes can overload that method. There's a small problem, though. On unarchiving all we have is a set of named byte streams. In C++ the class name (as written in the source code) has nothing to do with its actual type. Thus, we need establish a correspondence ourselves. To do so we maintain a `class_name' => function_pointer` table (see the `unarchive_table_t` class above). Every function in this table is supposed to create a new object of the `class_name'` type. The `archive_node` class uses that table to construct an object of correct type. Next it invokes `read_archive` virtual method of newly created object, which does the actual job.

Note: this approach is very simple-minded (it does not handle classes with same names from different namespaces, multiple inheritance, etc), but it happens to work surprisingly well.

#### 9.4.2.2 GINAC\_BIND\_UNARCHIVER

```
#define GINAC_BIND_UNARCHIVER(  
    classname)
```

**Value:**

```
classname ## _unarchiver::classname ## _unarchiver() \
{ \
    static GiNaC::unarchive_table_t table; \
    if (usecount++ == 0) { \
        table.insert(std::string(#classname), \
            &(classname ## _unarchiver::create)); \
    } \
} \
GiNaC::basic* classname ## _unarchiver::create() \
{ \
    return new classname(); \
} \
classname ## _unarchiver::~~ classname ## _unarchiver() { } \
int classname ## _unarchiver::usecount = 0
```

## 9.5 assertion.h File Reference

Assertion macro definition.

### Macros

- `#define GINAC_ASSERT(X)`  
Assertion macro for checking invariances.



[GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#),  
[GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::psi1\\_deriv\(\)](#),  
[GiNaC::psi2\\_deriv\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::matrix::rank\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::function::real\\_part\(\)](#),  
[GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::relational::return\\_type\(\)](#),  
[GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::relational::return\\_type\\_tinfo\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#),  
[GiNaC::function::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::indexed::simplify\\_indexed\(\)](#),  
[GiNaC::sin\\_deriv\(\)](#), [GiNaC::sinh\\_deriv\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#),  
[GiNaC::ex::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::swap\(\)](#), [GiNaC::tan\\_deriv\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_deriv\(\)](#),  
[GiNaC::tanh\\_series\(\)](#), [GiNaC::tgamma\\_deriv\(\)](#), [GiNaC::numeric::to\\_double\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), [GiNaC::numeric::to\\_long\(\)](#),  
[GiNaC::indexed::validate\(\)](#), [GiNaC::zeta1\\_deriv\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

## 9.6 basic.cpp File Reference

Implementation of [GiNaC](#)'s ABC.

```

#include "basic.h"
#include "ex.h"
#include "numeric.h"
#include "power.h"
#include "add.h"
#include "symbol.h"
#include "lst.h"
#include "ncmul.h"
#include "relational.h"
#include "operators.h"
#include "wildcard.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <typeinfo>

```

### Classes

- struct [GiNaC::evalf\\_map\\_function](#)  
*Function object to be applied by [basic::evalf\(\)](#).*
- struct [GiNaC::evalm\\_map\\_function](#)  
*Function object to be applied by [basic::evalm\(\)](#).*
- struct [GiNaC::eval\\_integ\\_map\\_function](#)  
*Function object to be applied by [basic::eval\\_integ\(\)](#).*
- struct [GiNaC::derivative\\_map\\_function](#)  
*Function object to be applied by [basic::derivative\(\)](#).*
- struct [GiNaC::expand\\_map\\_function](#)  
*Function object to be applied by [basic::expand\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([basic](#), void, [print\\_func](#)< [print\\_context](#)>(&[basic::do\\_print](#)), [print\\_func](#)< [print\\_tree](#)>(&[basic::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#)>(&[basic::do\\_print\\_python\\_repr](#))) [basic](#)

*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by [duplicate\(\)](#)), so it can copy the [tinfo\\_key](#) and the hash value.*

## Variables

- [GiNaC::evalm\\_map\\_function](#) [GiNaC::map\\_evalm](#)
- [GiNaC::eval\\_integ\\_map\\_function](#) [GiNaC::map\\_eval\\_integ](#)

### 9.6.1 Detailed Description

Implementation of [GiNaC](#)'s ABC.

## 9.7 basic.h File Reference

Interface to [GiNaC](#)'s ABC.

```
#include "flags.h"
#include "ptr.h"
#include "assertion.h"
#include "registrar.h"
#include <cstdint>
#include <map>
#include <set>
#include <typeinfo>
#include <vector>
#include <utility>
```

## Classes

- struct [GiNaC::map\\_function](#)  
*Function object for [map\(\)](#).*
- class [GiNaC::visitor](#)  
*Degenerate base class for visitors.*
- class [GiNaC::basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*

## Namespaces

- namespace [GiNaC](#)



## Typedefs

- typedef std::vector< [ex](#) > [GiNaC::exvector](#)
- typedef std::set< [ex](#), [ex\\_is\\_less](#) > [GiNaC::exset](#)
- typedef std::map< [ex](#), [ex](#), [ex\\_is\\_less](#) > [GiNaC::exmap](#)

## Functions

- template<class T>  
bool [GiNaC::is\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, including base classes.*
- template<class T>  
bool [GiNaC::is\\_exactly\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, not including base classes.*
- template<class B, typename... Args>  
B & [GiNaC::dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B>  
B & [GiNaC::dynallocate](#) (std::initializer\_list< [ex](#) > il)  
*Constructs a new (class basic or derived) B object on the heap.*

### 9.7.1 Detailed Description

Interface to [GiNaC](#)'s ABC.

## 9.8 class\_info.h File Reference

Helper templates to provide per-class information for class hierarchies.

```
#include <cstddef>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <stdexcept>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::class\\_info](#)< [OPT](#) >
- struct [GiNaC::class\\_info](#)< [OPT](#) >::tree\_node

## Namespaces

- namespace [GiNaC](#)

## Variables

- `template<class OPT>`  
`class_info< OPT > * GiNaC::class_info< OPT >::first = nullptr`
- `template<class OPT>`  
`bool GiNaC::class_info< OPT >::parents_identified = false`

### 9.8.1 Detailed Description

Helper templates to provide per-class information for class hierarchies.

## 9.9 clifford.cpp File Reference

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "clifford.h"
#include "ex.h"
#include "idx.h"
#include "ncmul.h"
#include "symbol.h"
#include "numeric.h"
#include "symmetry.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <algorithm>
#include <stdexcept>
```

## Classes

- struct [GiNaC::is\\_not\\_a\\_clifford](#)  
*Predicate for finding non-clifford objects.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`clifford`, `indexed`, `print_func< print_dflt >(&clifford::do_print_dflt)`, `print_func< print_latex >(&clifford::do_print_latex)`, `print_func< print_tree >(&clifford::do_print_tree)`) `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT`(`diracone`
- `GiNaC::print_func< print_dflt >(&diracone::do_print)`, `print_func< print_latex >(&diracone`
- `GiNaC::GINAC_BIND_UNARCHIVER` (`clifford`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`cliffordunit`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`diracone`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`diracgamma`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`diracgamma5`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`diracgammaL`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`diracgammaR`)
- static bool `GiNaC::is_dirac_slash` (const `ex` &seq0)
- static void `GiNaC::base_and_index` (const `ex` &c, `ex` &b, `ex` &i)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$ .*
- `ex` `GiNaC::dirac_ONE` (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned `GiNaC::get_dim_uint` (const `ex` &e)
- `ex` `GiNaC::clifford_unit` (const `ex` &mu, const `ex` &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- `ex` `GiNaC::dirac_gamma` (const `ex` &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- `ex` `GiNaC::dirac_gamma5` (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- `ex` `GiNaC::dirac_gammaL` (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- `ex` `GiNaC::dirac_gammaR` (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- `ex` `GiNaC::dirac_slash` (const `ex` &e, const `ex` &dim, unsigned char rl=0)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
- static unsigned char `GiNaC::get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- static `ex` `GiNaC::trace_string` (`exvector::const_iterator` ix, `size_t` num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- `ex` `GiNaC::dirac_trace` (const `ex` &e, const `std::set< unsigned char >` &rls, const `ex` &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- `ex` `GiNaC::dirac_trace` (const `ex` &e, const `lst` &rl, const `ex` &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- `ex` `GiNaC::dirac_trace` (const `ex` &e, unsigned char rl=0, const `ex` &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- `ex` `GiNaC::canonicalize_clifford` (const `ex` &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- `ex` `GiNaC::clifford_star_bar` (const `ex` &e, bool do\_bar, unsigned options)  
*An auxillary function performing `clifford_star()` and `clifford_bar()`.*
- `ex` `GiNaC::clifford_prime` (const `ex` &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- `ex` `GiNaC::remove_dirac_ONE` (const `ex` &e, unsigned char rl=0, unsigned options=0)  
*Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.*
- int `GiNaC::clifford_max_label` (const `ex` &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if `e` contains at least one, otherwise returns -1.*
- `ex` `GiNaC::clifford_norm` (const `ex` &e)

- Calculation of the norm in the Clifford algebra.*
- `ex GiNaC::clifford_inverse` (const `ex` &`e`)
- Calculation of the inverse in the Clifford algebra.*
- `ex GiNaC::lst_to_clifford` (const `ex` &`v`, const `ex` &`mu`, const `ex` &`metr`, unsigned char `rl=0`)
- List or vector conversion into the Clifford vector.*
- `ex GiNaC::lst_to_clifford` (const `ex` &`v`, const `ex` &`e`)
- List or vector conversion into the Clifford vector.*
- static `ex GiNaC::get_clifford_comp` (const `ex` &`e`, const `ex` &`c`, bool `root=true`)
- Auxiliary structure to define a function for stripping one Clifford unit from vectors.*
- `lst GiNaC::clifford_to_lst` (const `ex` &`e`, const `ex` &`c`, bool `algebraic=true`)
- An inverse function to `lst_to_clifford()`.*
- `ex GiNaC::clifford_moebius_map` (const `ex` &`a`, const `ex` &`b`, const `ex` &`c`, const `ex` &`d`, const `ex` &`v`, const `ex` &`G`, unsigned char `rl=0`)
- Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*
- `ex GiNaC::clifford_moebius_map` (const `ex` &`M`, const `ex` &`v`, const `ex` &`G`, unsigned char `rl=0`)
- The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

## Variables

- `GiNaC::tensor`

## 9.9.1 Detailed Description

Implementation of `GiNaC`'s clifford algebra (Dirac gamma) objects.

## 9.10 clifford.h File Reference

Interface to `GiNaC`'s clifford algebra (Dirac gamma) objects.

```
#include "indexed.h"
#include "tensor.h"
#include "symbol.h"
#include "idx.h"
#include <set>
```

## Classes

- class `GiNaC::clifford`  
*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class `GiNaC::diracone`  
*This class represents the Clifford algebra unity element.*
- class `GiNaC::cliffordunit`  
*This class represents the Clifford algebra generators (units).*
- class `GiNaC::diracgamma`  
*This class represents the Dirac gamma Lorentz vector.*
- class `GiNaC::diracgamma5`  
*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
- class `GiNaC::diracgammaL`  
*This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).*
- class `GiNaC::diracgammaR`  
*This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([clifford](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracone](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([cliffordunit](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgamma](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgamma5](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgammaL](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([diracgammaR](#))
- [bool GiNaC::is\\_clifford\\_tinfo](#) (const [return\\_type\\_t](#) &ti)
 

*Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).*
- [ex GiNaC::dirac\\_ONE](#) (unsigned char rl=0)
 

*Create a Clifford unity object.*
- [ex GiNaC::clifford\\_unit](#) (const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)
 

*Create a Clifford unit object.*
- [ex GiNaC::dirac\\_gamma](#) (const [ex](#) &mu, unsigned char rl=0)
 

*Create a Dirac gamma object.*
- [ex GiNaC::dirac\\_gamma5](#) (unsigned char rl=0)
 

*Create a Dirac gamma5 object.*
- [ex GiNaC::dirac\\_gammaL](#) (unsigned char rl=0)
 

*Create a Dirac gammaL object.*
- [ex GiNaC::dirac\\_gammaR](#) (unsigned char rl=0)
 

*Create a Dirac gammaR object.*
- [ex GiNaC::dirac\\_slash](#) (const [ex](#) &e, const [ex](#) &dim, unsigned char rl=0)
 

*Create a term of the form  $e_{\mu} * \gamma^{\sim\mu}$  with a unique index  $\mu$ .*
- [ex GiNaC::dirac\\_trace](#) (const [ex](#) &e, const std::set< unsigned char > &rls, const [ex](#) &trONE=4)
 

*Calculate dirac traces over the specified set of representation labels.*
- [ex GiNaC::dirac\\_trace](#) (const [ex](#) &e, const [lst](#) &rl, const [ex](#) &trONE=4)
 

*Calculate dirac traces over the specified list of representation labels.*
- [ex GiNaC::dirac\\_trace](#) (const [ex](#) &e, unsigned char rl=0, const [ex](#) &trONE=4)
 

*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- [ex GiNaC::canonicalize\\_clifford](#) (const [ex](#) &e)
 

*Bring all products of clifford objects in an expression into a canonical order.*
- [ex GiNaC::clifford\\_prime](#) (const [ex](#) &e)
 

*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- [ex GiNaC::clifford\\_star\\_bar](#) (const [ex](#) &e, bool do\_bar, unsigned options)
 

*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- [ex GiNaC::clifford\\_bar](#) (const [ex](#) &e)
 

*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- [ex GiNaC::clifford\\_star](#) (const [ex](#) &e)
 

*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- [ex GiNaC::remove\\_dirac\\_ONE](#) (const [ex](#) &e, unsigned char rl=0, unsigned options=0)
 

*Replaces [dirac\\_ONE](#)'s (with a [representation\\_label](#) no less than  $rl$ ) in  $e$  with 1.*
- [int GiNaC::clifford\\_max\\_label](#) (const [ex](#) &e, bool ignore\_ONE=false)
 

*Returns the maximal representation label of a clifford object if  $e$  contains at least one, otherwise returns -1.*
- [ex GiNaC::clifford\\_norm](#) (const [ex](#) &e)

- Calculation of the norm in the Clifford algebra.*

  - `ex GiNaC::clifford_inverse` (const `ex` &e)
- Calculation of the inverse in the Clifford algebra.*

  - `ex GiNaC::lst_to_clifford` (const `ex` &v, const `ex` &mu, const `ex` &metr, unsigned char rl=0)
- List or vector conversion into the Clifford vector.*

  - `ex GiNaC::lst_to_clifford` (const `ex` &v, const `ex` &e)
- List or vector conversion into the Clifford vector.*

  - `lst GiNaC::clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)
- An inverse function to `lst_to_clifford()`.*

  - `ex GiNaC::clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)
- Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.*

  - `ex GiNaC::clifford_moebius_map` (const `ex` &M, const `ex` &v, const `ex` &G, unsigned char rl=0)
- The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

### 9.10.1 Detailed Description

Interface to `GiNaC`'s clifford algebra (Dirac gamma) objects.

## 9.11 color.cpp File Reference

Implementation of `GiNaC`'s color (SU(3) Lie algebra) objects.

```
#include "color.h"
#include "idx.h"
#include "ncmul.h"
#include "symmetry.h"
#include "operators.h"
#include "numeric.h"
#include "mul.h"
#include "power.h"
#include "symbol.h"
#include "archive.h"
#include "utils.h"
#include <stdexcept>
```

### Namespaces

- namespace `GiNaC`

### Macros

- `#define TEST_PERMUTATION(A, B, C, P)`
- `#define CMPINDICES(A, B, C)`

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`su3one`, `tensor`, `print_func< print_dflt >(&su3one::do_print)`, `print_func< print_latex >(&su3one::do_print_latex)`) `GINAC_IMPLEMENT_↵`  
`REGISTERED_CLASS_OPT(su3t`
- `GiNaC::print_func< print_dflt > (&su3t::do_print)`. `print_func< print_latex >(&su3t`
- `GiNaC::GINAC_BIND_UNARCHIVER` (`color`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3one`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3t`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3f`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3d`)
- static `ex` `GiNaC::permute_free_index_to_front` (const `exvector` &iv3, const `exvector` &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- `ex` `GiNaC::color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex` `GiNaC::color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex` `GiNaC::color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex` `GiNaC::color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex` `GiNaC::color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- static bool `GiNaC::is_color_tinfo` (const `return_type_t` &ti)  
*Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).*
- static unsigned char `GiNaC::get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- `ex` `GiNaC::color_trace` (const `ex` &e, const `std::set< unsigned char >` &rls)  
*Calculate color traces over the specified set of representation labels.*
- `ex` `GiNaC::color_trace` (const `ex` &e, const `lst` &rlI)  
*Calculate color traces over the specified list of representation labels.*
- `ex` `GiNaC::color_trace` (const `ex` &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 9.11.1 Detailed Description

Implementation of `GiNaC`'s color (SU(3) Lie algebra) objects.

### 9.11.2 Macro Definition Documentation

#### 9.11.2.1 TEST\_PERMUTATION

```
#define TEST_PERMUTATION(
    A,
    B,
    C,
    P)
```

#### Value:

```
if (iv3[B].is_equal(iv2[0]) && iv3[C].is_equal(iv2[1])) { \
    sig = P; \
    return iv3[A]; \
}
```

Referenced by `GiNaC::permute_free_index_to_front()`.

### 9.11.2.2 CMPINDICES

```
#define CMPINDICES(
    A,
    B,
    C)
```

#### Value:

```
((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))
```

Referenced by [GiNaC::su3d::eval\\_indexed\(\)](#), and [GiNaC::su3f::eval\\_indexed\(\)](#).

## 9.12 color.h File Reference

Interface to [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "indexed.h"
#include "tensor.h"
#include <set>
```

### Classes

- class [GiNaC::color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of SU(3), as used for calculations in quantum chromodynamics.*
- class [GiNaC::su3one](#)  
*This class represents the su(3) unity element.*
- class [GiNaC::su3t](#)  
*This class represents an su(3) generator.*
- class [GiNaC::su3f](#)  
*This class represents the tensor of antisymmetric su(3) structure constants.*
- class [GiNaC::su3d](#)  
*This class represents the tensor of symmetric su(3) structure constants.*

### Namespaces

- namespace [GiNaC](#)



## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER (color)`
- `GiNaC::GINAC_DECLARE_UNARCHIVER (su3one)`
- `GiNaC::GINAC_DECLARE_UNARCHIVER (su3t)`
- `GiNaC::GINAC_DECLARE_UNARCHIVER (su3f)`
- `GiNaC::GINAC_DECLARE_UNARCHIVER (su3d)`
- `ex GiNaC::color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex GiNaC::color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex GiNaC::color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex GiNaC::color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex GiNaC::color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- `ex GiNaC::color_trace` (const `ex` &e, const `std::set< unsigned char >` &rls)  
*Calculate color traces over the specified set of representation labels.*
- `ex GiNaC::color_trace` (const `ex` &e, const `lst` &rl)  
*Calculate color traces over the specified list of representation labels.*
- `ex GiNaC::color_trace` (const `ex` &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 9.12.1 Detailed Description

Interface to `GiNaC`'s color (SU(3) Lie algebra) objects.

## 9.13 compiler.h File Reference

Definition of optimizing macros.

### Macros

- `#define unlikely(cond)`
- `#define likely(cond)`
- `#define attribute_pure`

### 9.13.1 Detailed Description

Definition of optimizing macros.

## 9.13.2 Macro Definition Documentation

### 9.13.2.1 unlikely

```
#define unlikely(  
    cond)
```

**Value:**

(cond)

Referenced by [GiNaC::add::eval\(\)](#), [GiNaC::mul::eval\(\)](#), and [GiNaC::expairseq::evalchildren\(\)](#).

### 9.13.2.2 likely

```
#define likely(  
    cond)
```

**Value:**

(cond)

Referenced by [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::eval\(\)](#), and [GiNaC::power::eval\(\)](#).

### 9.13.2.3 attribute\_pure

```
#define attribute_pure
```

## 9.14 constant.cpp File Reference

Implementation of [GiNaC](#)'s constant types and some special constants.

```
#include "constant.h"  
#include "numeric.h"  
#include "ex.h"  
#include "archive.h"  
#include "utils.h"  
#include "inifcns.h"  
#include <stdexcept>  
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (constant, basic, print\_func< [print\\_context](#) >(&constant::do\_print). print\_func< [print\\_latex](#) >(&constant::do\_print\_latex). print\_func< [print\\_tree](#) >(&constant::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&constant::do\_print\_python\_repr)) constant
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (constant)

## Variables

- const [constant GiNaC::Pi](#) ("Pi", [PiEvalf](#), "\\pi", [domain::positive](#))  
*Pi.*
- const [constant GiNaC::Euler](#) ("Euler", [EulerEvalf](#), "\\gamma\_E", [domain::positive](#))  
*Euler's constant.*
- const [constant GiNaC::Catalan](#) ("Catalan", [CatalanEvalf](#), "G", [domain::positive](#))  
*Catalan's constant.*

### 9.14.1 Detailed Description

Implementation of [GiNaC](#)'s constant types and some special constants.

## 9.15 constant.h File Reference

Interface to [GiNaC](#)'s constant types and some special constants.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <string>
```

## Classes

- class [GiNaC::constant](#)  
*This class holds constants, symbols with specific numerical value.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef [ex](#)(\* [GiNaC::evalffunctype](#)) ()

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([constant](#))

### 9.15.1 Detailed Description

Interface to [GiNaC](#)'s constant types and some special constants.

## 9.16 container.h File Reference

Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include "ex.h"
#include "print.h"
#include "archive.h"
#include "assertion.h"
#include "compiler.h"
#include <algorithm>
#include <iterator>
#include <list>
#include <memory>
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::container\\_storage< C >](#)  
*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
- class [GiNaC::container< C >](#)  
*Wrapper template for making [GiNaC](#) classes out of STL containers.*

### Namespaces

- namespace [GiNaC](#)

### 9.16.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of STL containers.

## 9.17 crc32.h File Reference

CRC32 hash function.

### Namespaces

- namespace [GiNaC](#)

### Functions

- static unsigned [GiNaC::crc32](#) (const char \*c, const unsigned len, const unsigned crcinit)

### Variables

- static unsigned const [GiNaC::crctab](#) [256]

### 9.17.1 Detailed Description

CRC32 hash function.

Shamelessly stolen from GNU coreutils (cksum.c).

## 9.18 ex.cpp File Reference

Implementation of [GiNaC](#)'s light-weight expression handles.

```
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "matrix.h"
#include "power.h"
#include "lst.h"
#include "relational.h"
#include "utils.h"
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### 9.18.1 Detailed Description

Implementation of [GiNaC](#)'s light-weight expression handles.

## 9.19 ex.h File Reference

Interface to [GiNaC](#)'s light-weight expression handles.

```
#include "basic.h"
#include "ptr.h"
#include "compiler.h"
#include <functional>
#include <iosfwd>
#include <iterator>
#include <memory>
#include <stack>
```

## Classes

- class [GiNaC::library\\_init](#)  
*Helper class to initialize the library.*
- class [GiNaC::ex](#)  
*Lightweight wrapper for [GiNaC](#)'s symbolic objects.*
- class [GiNaC::const\\_iterator](#)
- struct [GiNaC::internal::\\_iter\\_rep](#)
- class [GiNaC::const\\_preorder\\_iterator](#)
- class [GiNaC::const\\_postorder\\_iterator](#)
- struct [GiNaC::ex\\_is\\_less](#)
- struct [GiNaC::ex\\_is\\_equal](#)
- struct [GiNaC::op0\\_is\\_equal](#)
- struct [GiNaC::ex\\_swap](#)
- class [GiNaC::pointer\\_to\\_map\\_function](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#)
- struct [std::hash< GiNaC::ex >](#)  
*Specialization of `std::hash()` for `ex` objects.*
- struct [std::equal\\_to< GiNaC::ex >](#)  
*Specialization of `std::equal_to()` for `ex` objects.*

## Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::internal](#)
- namespace [std](#)

## Functions

- bool [GiNaC::are\\_ex\\_trivially\\_equal](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exvector](#) &e)
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exset](#) &e)
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exmap](#) &e)
- [size\\_t GiNaC::nops](#) (const [ex](#) &thisex)
- [ex GiNaC::expand](#) (const [ex](#) &thisex, unsigned options=0)
- [ex GiNaC::conjugate](#) (const [ex](#) &thisex)
- [ex GiNaC::real\\_part](#) (const [ex](#) &thisex)
- [ex GiNaC::imag\\_part](#) (const [ex](#) &thisex)
- bool [GiNaC::has](#) (const [ex](#) &thisex, const [ex](#) &pattern, unsigned options=0)
- bool [GiNaC::find](#) (const [ex](#) &thisex, const [ex](#) &pattern, [exset](#) &found)
- bool [GiNaC::is\\_polynomial](#) (const [ex](#) &thisex, const [ex](#) &vars)
- int [GiNaC::degree](#) (const [ex](#) &thisex, const [ex](#) &s)
- int [GiNaC::ldegree](#) (const [ex](#) &thisex, const [ex](#) &s)
- [ex GiNaC::coeff](#) (const [ex](#) &thisex, const [ex](#) &s, int n=1)
- [ex GiNaC::numer](#) (const [ex](#) &thisex)

- `ex GiNaC::denom` (const `ex` &thisex)
- `ex GiNaC::numer_denom` (const `ex` &thisex)
- `ex GiNaC::normal` (const `ex` &thisex)
- `ex GiNaC::to_rational` (const `ex` &thisex, `exmap` &repl)
- `ex GiNaC::to_polynomial` (const `ex` &thisex, `exmap` &repl)
- `ex GiNaC::collect` (const `ex` &thisex, const `ex` &s, bool distributed=false)
- `ex GiNaC::eval` (const `ex` &thisex)
- `ex GiNaC::evalf` (const `ex` &thisex)
- `ex GiNaC::evalm` (const `ex` &thisex)
- `ex GiNaC::eval_integ` (const `ex` &thisex)
- `ex GiNaC::diff` (const `ex` &thisex, const `symbol` &s, unsigned nth=1)
- `ex GiNaC::series` (const `ex` &thisex, const `ex` &r, int order, unsigned options=0)
- bool `GiNaC::match` (const `ex` &thisex, const `ex` &pattern, `exmap` &repl\_lst)
- `ex GiNaC::simplify_indexed` (const `ex` &thisex, unsigned options=0)
- `ex GiNaC::simplify_indexed` (const `ex` &thisex, const `scalar_products` &sp, unsigned options=0)
- `ex GiNaC::symmetrize` (const `ex` &thisex)
- `ex GiNaC::symmetrize` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::antisymmetrize` (const `ex` &thisex)
- `ex GiNaC::antisymmetrize` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::symmetrize_cyclic` (const `ex` &thisex)
- `ex GiNaC::symmetrize_cyclic` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::op` (const `ex` &thisex, size\_t i)
- `ex GiNaC::lhs` (const `ex` &thisex)
- `ex GiNaC::rhs` (const `ex` &thisex)
- bool `GiNaC::is_zero` (const `ex` &thisex)
- void `GiNaC::swap` (`ex` &e1, `ex` &e2)
- `ex GiNaC::subs` (const `ex` &thisex, const `exmap` &m, unsigned options=0)
- `ex GiNaC::subs` (const `ex` &thisex, const `lst` &ls, const `lst` &lr, unsigned options=0)
- `ex GiNaC::subs` (const `ex` &thisex, const `ex` &e, unsigned options=0)
- template<class T>  
bool `GiNaC::is_a` (const `ex` &obj)  
*Check if ex is a handle to a T, including base classes.*
- template<class T>  
bool `GiNaC::is_exactly_a` (const `ex` &obj)  
*Check if ex is a handle to a T, not including base classes.*
- template<class T>  
`attribute_pure` const T &`GiNaC::ex_to` (const `ex` &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<> void `std::swap` (`GiNaC::ex` &a, `GiNaC::ex` &b)  
*Specialization of `std::swap()` for ex objects.*

## Variables

- static `library_init` `GiNaC::library_initializer`  
*For construction of flyweights, etc.*
- const `basic` \* `GiNaC::_num0_bp`

## 9.19.1 Detailed Description

Interface to `GiNaC`'s light-weight expression handles.

## 9.20 excompiler.cpp File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "excompiler.h"
#include "ex.h"
#include "lst.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include <cstdlib>
#include <fstream>
#include <ios>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym, [FUNCP\\_1P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym1, const [symbol](#) &sym2, [FUNCP\\_2P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [lst](#) &exprs, const [lst](#) &syms, [FUNCP\\_CUBA](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_1P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_1P](#) to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_2P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_2P](#) to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_CUBA](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_CUBA](#) to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 9.20.1 Detailed Description

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

## 9.21 excompiler.h File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "lst.h"
#include <string>
```



## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef double(\* [GiNaC::FUNCP\\_1P](#)) (double)  
*Function pointer with one function parameter.*
- typedef double(\* [GiNaC::FUNCP\\_2P](#)) (double, double)  
*Function pointer with two function parameters.*
- typedef void(\* [GiNaC::FUNCP\\_CUBA](#)) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).*

## Functions

- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym, [FUNCP\\_1P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym1, const [symbol](#) &sym2, [FUNCP\\_2P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [lst](#) &exprs, const [lst](#) &syms, [FUNCP\\_CUBA](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_1P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_1P](#) to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_2P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_2P](#) to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_CUBA](#) &fp)  
*Opens an existing so-file and returns a function pointer of type [FUNCP\\_CUBA](#) to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 9.21.1 Detailed Description

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

## 9.22 expair.cpp File Reference

Implementation of expression pairs (building blocks of `expairseq`).

```
#include "expair.h"
#include "operators.h"
```

## Namespaces

- namespace [GiNaC](#)

### 9.22.1 Detailed Description

Implementation of expression pairs (building blocks of `expairseq`).

## 9.23 `expair.h` File Reference

Definition of expression pairs (building blocks of `expairseq`).

```
#include "ex.h"
#include "numeric.h"
#include "print.h"
```

### Classes

- class [GiNaC::expair](#)  
*A pair of expressions.*
- struct [GiNaC::expair\\_is\\_less](#)  
*Function object for insertion into third argument of STL's `sort()` etc.*
- struct [GiNaC::expair\\_rest\\_is\\_less](#)  
*Function object not caring about the numerical coefficients for insertion into third argument of STL's `sort()`.*
- struct [GiNaC::expair\\_swap](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- void [GiNaC::swap](#) ([expair](#) &e1, [expair](#) &e2)

### 9.23.1 Detailed Description

Definition of expression pairs (building blocks of `expairseq`).

## 9.24 expairseq.cpp File Reference

Implementation of sequences of expression pairs.

```
#include "expairseq.h"
#include "lst.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "wildcard.h"
#include "archive.h"
#include "operators.h"
#include "utils.h"
#include "hash_seed.h"
#include "indexed.h"
#include "compiler.h"
#include <algorithm>
#include <iostream>
#include <iterator>
#include <memory>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([expairseq](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[expairseq::do\\_print](#)). [print\\_func](#)< [print\\_tree](#) >(&[expairseq::do\\_print\\_tree](#))) [class](#) [epp\\_is\\_less](#)
- [epvector](#) \* [GiNaC::conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*

### 9.24.1 Detailed Description

Implementation of sequences of expression pairs.

## 9.25 expairseq.h File Reference

Interface to sequences of expression pairs.

```
#include "expair.h"
#include "indexed.h"
#include <vector>
```

## Classes

- class [GiNaC::expairseq](#)  
*A sequence of class expair.*
- class [GiNaC::make\\_flat\\_inserter](#)  
*Class to handle the renaming of dummy indices.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< [expair](#) > [GiNaC::epvector](#)  
*expair-vector*
- typedef epvector::iterator [GiNaC::epp](#)  
*expair-vector pointer*

## Functions

- [epvector](#) \* [GiNaC::conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*

### 9.25.1 Detailed Description

Interface to sequences of expression pairs.

## 9.26 exprseq.cpp File Reference

Implementation of [GiNaC](#)'s exprseq.

```
#include "exprseq.h"
```

## Namespaces

- namespace [GiNaC](#)

### 9.26.1 Detailed Description

Implementation of [GiNaC](#)'s exprseq.

## 9.27 exprseq.h File Reference

Definition of [GiNaC](#)'s exprseq.

```
#include "container.h"  
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef [container](#)< std::vector > [GiNaC::exprseq](#)

### 9.27.1 Detailed Description

Definition of [GiNaC](#)'s exprseq.

## 9.28 factor.cpp File Reference

Polynomial factorization (implementation).

```
#include "factor.h"  
#include "ex.h"  
#include "numeric.h"  
#include "operators.h"  
#include "inifcns.h"  
#include "symbol.h"  
#include "relational.h"  
#include "power.h"  
#include "mul.h"  
#include "normal.h"  
#include "add.h"  
#include <type_traits>  
#include <algorithm>  
#include <limits>  
#include <list>  
#include <vector>  
#include <stack>  
#include <cln/cln.h>
```

### Namespaces

- namespace [GiNaC](#)

## Macros

- `#define DCOUT(str)`
- `#define DCOUTVAR(var)`
- `#define DCOUT2(str, var)`
- `#define USE_SAME_DEGREE_FACTOR`

## Functions

- `ex GiNaC::factor` (const `ex` &poly, unsigned options)

*Interface function to the outside world.*

### 9.28.1 Detailed Description

Polynomial factorization (implementation).

The interface function `factor()` at the end of this file is defined in the `GiNaC` namespace. All other utility functions and classes are defined in an additional anonymous namespace.

Factorization starts by doing a square free factorization and making the coefficients integer. Then, depending on the number of free variables it proceeds either in dedicated univariate or multivariate factorization code.

Univariate factorization does a modular factorization via Berlekamp's algorithm and distinct degree factorization. Hensel lifting is used at the end.

Multivariate factorization uses the univariate factorization (applying a evaluation homomorphism first) and Hensel lifting raises the answer to the multivariate domain. The Hensel lifting code is completely distinct from the code used by the univariate factorization.

Algorithms used can be found in [Wan] An Improved Multivariate Polynomial Factoring Algorithm, P.S.Wang, Mathematics of Computation, Vol. 32, No. 144 (1978) 1215–1231. [GCL] Algorithms for Computer Algebra, K.O.Geddes, S.R.Czapor, G.Labahn, Springer Verlag, 1992. [Mig] Some Useful Bounds, M.Mignotte, In "Computer Algebra, Symbolic and Algebraic Computation" (B.Buchberger et al., eds.), pp. 259-263, Springer-Verlag, New York, 1982.

### 9.28.2 Macro Definition Documentation

#### 9.28.2.1 DCOUT

```
#define DCOUT(  
    str)
```

#### 9.28.2.2 DCOUTVAR

```
#define DCOUTVAR(  
    var)
```

#### 9.28.2.3 DCOUT2

```
#define DCOUT2(  
    str,  
    var)
```

### 9.28.2.4 USE\_SAME\_DEGREE\_FACTOR

```
#define USE_SAME_DEGREE_FACTOR
```

## 9.29 factor.h File Reference

Polynomial factorization.

### Namespaces

- namespace [GiNaC](#)

### Functions

- [ex GiNaC::factor](#) (const [ex](#) &poly, unsigned options)  
*Interface function to the outside world.*

### 9.29.1 Detailed Description

Polynomial factorization.

## 9.30 fail.cpp File Reference

Implementation of class signaling failure of operation.

```
#include "fail.h"  
#include "archive.h"  
#include "utils.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (fail, [basic](#), [print\\_func< print\\_context >\(&fail::do\\_print\)](#), [print\\_func< print\\_tree >\(&fail::do\\_print\\_tree\)](#)) [GINAC\\_BIND\\_UNARCHIVER\(fail\)](#)

### 9.30.1 Detailed Description

Implementation of class signaling failure of operation.

Considered somewhat obsolete (most of this can be replaced by exceptions).

## 9.31 fail.h File Reference

Interface to class signaling failure of operation.

```
#include "basic.h"  
#include "archive.h"
```

### Classes

- class [GiNaC::fail](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([fail](#))

### 9.31.1 Detailed Description

Interface to class signaling failure of operation.

Considered obsolete somewhat obsolete (most of this can be replaced by exceptions).

## 9.32 fderivative.cpp File Reference

Implementation of abstract derivatives of functions.

```
#include "fderivative.h"  
#include "operators.h"  
#include "archive.h"  
#include "utils.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fderivative](#), [function](#), [print\\_func< print\\_context >\(&fderivative::do\\_print\)](#), [print\\_func< print\\_latex >\(&fderivative::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&fderivative::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&fderivative::do\\_print\\_tree\)](#)) [fderivative](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([fderivative](#))



### 9.32.1 Detailed Description

Implementation of abstract derivatives of functions.

## 9.33 fderivative.h File Reference

Interface to abstract derivatives of functions.

```
#include "function.h"  
#include <set>
```

### Classes

- class [GiNaC::fderivative](#)  
*This class represents the (abstract) derivative of a symbolic function.*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::multiset< unsigned > [GiNaC::paramset](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([fderivative](#))

### 9.33.1 Detailed Description

Interface to abstract derivatives of functions.

## 9.34 flags.h File Reference

Collection of all flags used through the [GiNaC](#) framework.

## Classes

- class [GiNaC::expand\\_options](#)  
*Flags to control the behavior of `expand()`.*
- class [GiNaC::has\\_options](#)  
*Flags to control the behavior of `has()`.*
- class [GiNaC::subs\\_options](#)  
*Flags to control the behavior of `subs()`.*
- class [GiNaC::domain](#)  
*Domain of an object.*
- class [GiNaC::series\\_options](#)  
*Flags to control series expansion.*
- class [GiNaC::determinant\\_algo](#)  
*Switch to control algorithm for determinant computation.*
- class [GiNaC::solve\\_algo](#)  
*Switch to control algorithm for linear system solving.*
- class [GiNaC::status\\_flags](#)  
*Flags to store information about the state of an object.*
- class [GiNaC::info\\_flags](#)  
*Possible attributes an object can have.*
- class [GiNaC::return\\_types](#)
- class [GiNaC::remember\\_strategies](#)  
*Strategies how to clean up the function remember cache.*
- class [GiNaC::factor\\_options](#)  
*Flags to control the polynomial factorization.*

## Namespaces

- namespace [GiNaC](#)

### 9.34.1 Detailed Description

Collection of all flags used through the [GiNaC](#) framework.

## 9.35 `function.cpp` File Reference

Implementation of class of symbolic functions.

```
#include "function.h"
#include "operators.h"
#include "fderivative.h"
#include "ex.h"
#include "lst.h"
#include "symmetry.h"
#include "print.h"
#include "power.h"
#include "archive.h"
#include "inifcns.h"
#include "utils.h"
```

```
#include "hash_seed.h"
#include "remember.h"
#include <iostream>
#include <limits>
#include <list>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (function)

### 9.35.1 Detailed Description

Implementation of class of symbolic functions.

## 9.36 function.h File Reference

Interface to class of symbolic functions.

```
#include "exprseq.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::function\\_options](#)
- class [GiNaC::do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*

- class [GiNaC::function](#)

*The class function is used to implement builtin functions like sin, cos... and user defined functions.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define DECLARE_FUNCTION_1P(NAME)`
- `#define DECLARE_FUNCTION_2P(NAME)`
- `#define DECLARE_FUNCTION_3P(NAME)`
- `#define DECLARE_FUNCTION_4P(NAME)`
- `#define DECLARE_FUNCTION_5P(NAME)`
- `#define DECLARE_FUNCTION_6P(NAME)`
- `#define DECLARE_FUNCTION_7P(NAME)`
- `#define DECLARE_FUNCTION_8P(NAME)`
- `#define DECLARE_FUNCTION_9P(NAME)`
- `#define DECLARE_FUNCTION_10P(NAME)`
- `#define DECLARE_FUNCTION_11P(NAME)`
- `#define DECLARE_FUNCTION_12P(NAME)`
- `#define DECLARE_FUNCTION_13P(NAME)`
- `#define DECLARE_FUNCTION_14P(NAME)`
- `#define REGISTER_FUNCTION(NAME, OPT)`
- `#define is_ex_the_function(OBJ, FUNCNAME)`

## Typedefs

- `typedef ex(* GiNaC::eval_funcp) ()`
- `typedef ex(* GiNaC::evalf_funcp) ()`
- `typedef ex(* GiNaC::conjugate_funcp) ()`
- `typedef ex(* GiNaC::real_part_funcp) ()`
- `typedef ex(* GiNaC::imag_part_funcp) ()`
- `typedef ex(* GiNaC::expand_funcp) ()`
- `typedef ex(* GiNaC::derivative_funcp) ()`
- `typedef ex(* GiNaC::expl_derivative_funcp) ()`
- `typedef ex(* GiNaC::power_funcp) ()`
- `typedef ex(* GiNaC::series_funcp) ()`
- `typedef void(* GiNaC::print_funcp) ()`
- `typedef bool(* GiNaC::info_funcp) ()`
- `typedef ex(* GiNaC::eval_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::evalf_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::real_part_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)`
- `typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)`
- `typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)`
- `typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)`
- `typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)`
- `typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)`
- `typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)`
- `typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)`

- [illegible]



- [illegible]







- [illegible]

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (function)
- `template<typename T>`  
`bool GiNaC::is\_the\_function (const ex &x)`

### 9.36.1 Detailed Description

Interface to class of symbolic functions.

### 9.36.2 Macro Definition Documentation

#### 9.36.2.1 DECLARE\_FUNCTION\_1P

```
#define DECLARE_FUNCTION_1P(  
    NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 1; \
template< typename T1 > const GiNaC::function NAME( const T1 & p1 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1) ); \
}
```

#### 9.36.2.2 DECLARE\_FUNCTION\_2P

```
#define DECLARE_FUNCTION_2P(  
    NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 2; \
template< typename T1, typename T2 > const GiNaC::function NAME( const T1 & p1, const T2 & p2 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2) ); \
}
```

#### 9.36.2.3 DECLARE\_FUNCTION\_3P

```
#define DECLARE_FUNCTION_3P(  
    NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 3; \
template< typename T1, typename T2, typename T3 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, \
    const T3 & p3 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3) ); \
}
```

## 9.36.2.4 DECLARE\_FUNCTION\_4P

```
#define DECLARE_FUNCTION_4P (
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 4; \
template< typename T1, typename T2, typename T3, typename T4 > const GiNaC::function NAME( const T1 & p1,
    const T2 & p2, const T3 & p3, const T4 & p4 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4)
    ); \
}
```

## 9.36.2.5 DECLARE\_FUNCTION\_5P

```
#define DECLARE_FUNCTION_5P (
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 5; \
template< typename T1, typename T2, typename T3, typename T4, typename T5 > const GiNaC::function NAME(
    const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5) ); \
}
```

## 9.36.2.6 DECLARE\_FUNCTION\_6P

```
#define DECLARE_FUNCTION_6P (
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 6; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
    T6 & p6 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6) ); \
}
```

## 9.36.2.7 DECLARE\_FUNCTION\_7P

```
#define DECLARE_FUNCTION_7P (
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 7; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
    T6 & p6, const T7 & p7 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7) ); \
}
```

### 9.36.2.8 DECLARE\_FUNCTION\_8P

```
#define DECLARE_FUNCTION_8P (
    NAME)
```

#### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 8; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4,
    const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8) ); \
}
```

### 9.36.2.9 DECLARE\_FUNCTION\_9P

```
#define DECLARE_FUNCTION_9P (
    NAME)
```

#### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 9; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3,
    const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9) ); \
}
```

### 9.36.2.10 DECLARE\_FUNCTION\_10P

```
#define DECLARE_FUNCTION_10P (
    NAME)
```

#### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 10; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10 > const GiNaC::function NAME( const T1 & p1, const T2 & p2,
    const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 &
    p9, const T10 & p10 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
    GiNaC::ex(p10) ); \
}
```

### 9.36.2.11 DECLARE\_FUNCTION\_11P

```
#define DECLARE_FUNCTION_11P (
    NAME)
```

#### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 11; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11 > const GiNaC::function NAME( const T1 & p1,
    const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 &
    p8, const T9 & p9, const T10 & p10, const T11 & p11 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
    GiNaC::ex(p10), GiNaC::ex(p11) ); \
}
```

## 9.36.2.12 DECLARE\_FUNCTION\_12P

```
#define DECLARE_FUNCTION_12P(
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 12; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12 > const GiNaC::function NAME( const
    T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7,
    const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 & p12 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
    GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12) ); \
}
```

## 9.36.2.13 DECLARE\_FUNCTION\_13P

```
#define DECLARE_FUNCTION_13P(
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 13; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
    T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &
    p12, const T13 & p13 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
    GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13) ); \
}
```

## 9.36.2.14 DECLARE\_FUNCTION\_14P

```
#define DECLARE_FUNCTION_14P(
    NAME)
```

## Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 14; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
    T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &
    p12, const T13 & p13, const T14 & p14 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
    GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13), GiNaC::ex(p14) ); \
}
```

## 9.36.2.15 REGISTER\_FUNCTION

```
#define REGISTER_FUNCTION(
    NAME,
    OPT)
```

## Value:

```
unsigned NAME##_SERIAL::serial = \
    GiNaC::function::register_new(GiNaC::function_options(#NAME, NAME##_NPARAMS).OPT);
```

### 9.36.2.16 is\_ex\_the\_function

```
#define is_ex_the_function(  
    OBJ,  
    FUNCNAME)
```

#### Value:

(GiNaC::is\_the\_function<FUNCNAME##\_SERIAL> (OBJ) )

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::tan\\_eval\(\)](#), and [GiNaC::tanh\\_eval\(\)](#).

## 9.37 ginac.h File Reference

This include file includes all other public [GiNaC](#) headers.

```
#include "version.h"  
#include "basic.h"  
#include "ex.h"  
#include "normal.h"  
#include "archive.h"  
#include "print.h"  
#include "constant.h"  
#include "fail.h"  
#include "integral.h"  
#include "lst.h"  
#include "matrix.h"  
#include "numeric.h"  
#include "power.h"  
#include "relational.h"  
#include "structure.h"  
#include "symbol.h"  
#include "pseries.h"  
#include "wildcard.h"  
#include "symmetry.h"  
#include "expair.h"  
#include "expairseq.h"  
#include "add.h"  
#include "mul.h"  
#include "exprseq.h"  
#include "function.h"  
#include "ncmul.h"  
#include "inifcns.h"  
#include "fderivative.h"  
#include "operators.h"  
#include "hash_map.h"  
#include "idx.h"  
#include "indexed.h"  
#include "tensor.h"  
#include "color.h"  
#include "clifford.h"  
#include "factor.h"  
#include "integration_kernel.h"  
#include "excompiler.h"  
#include "Gt.h"  
#include "parser.h"
```

### 9.37.1 Detailed Description

This include file includes all other public [GiNaC](#) headers.

## 9.38 Gt.cpp File Reference

Implementation of [GiNaC](#)'s elliptic multiple polylogarithms.

```
#include <type_traits>
#include <sstream>
#include <limits>
#include <cln/cln.h>
#include "add.h"
#include "mul.h"
#include "power.h"
#include "lst.h"
#include "matrix.h"
#include "symbol.h"
#include "numeric.h"
#include "inifcns.h"
#include "operators.h"
#include "relational.h"
#include "utils_multi_iterator.h"
#include "hash_seed.h"
#include "constant.h"
#include "Gt.h"
#include "Gt_helpers.h"
```

### Namespaces

- namespace [GiNaC](#)

### Macros

- `#define DEBUG\_GT 0`

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([Gt](#), [basic](#), [print\\_func](#)< [print\\_dflt](#) >(&[Gt::do\\_print](#)). [print\\_func](#)< [print\\_python](#) >(&[Gt::do\\_print](#)). [print\\_func](#)< [print\\_latex](#) >(&[Gt::do\\_print\\_latex](#))) [Gt](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([Gt](#))

### 9.38.1 Detailed Description

Implementation of [GiNaC](#)'s elliptic multiple polylogarithms.

## 9.38.2 Macro Definition Documentation

### 9.38.2.1 DEBUG\_GT

```
#define DEBUG_GT 0
```

Referenced by [GiNaC::Gt::decomposeIntegrationPath\(\)](#), [GiNaC::Gt::ex\\_evaluate\(\)](#), [GiNaC::Gt::ex\\_prepare\(\)](#), [GiNaC::Gt::lst\\_evaluate\(\)](#), [GiNaC::Gt::qExpand\(\)](#), [GiNaC::Gt::regularise\(\)](#), [GiNaC::Gt::tauToFundamental\(\)](#), and [GiNaC::Gt::zisToFundamental\(\)](#).

## 9.39 Gt.h File Reference

Interface to [GiNaC](#)'s elliptic multiple polylogarithms.

```
#include "archive.h"
#include "basic.h"
#include "ex.h"
```

### Classes

- class [GiNaC::Gt](#)  
*Elliptic Multiple Polylogarithm.*
- struct [GiNaC::Gt::kernel](#)

### Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::Gt\\_detail](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Gt)

### 9.39.1 Detailed Description

Interface to [GiNaC](#)'s elliptic multiple polylogarithms.

## 9.40 Gt\_helpers.cpp File Reference

Implementation of helper functions for elliptic multiple polylogarithms.

```
#include "add.h"
#include "mul.h"
#include "power.h"
#include "lst.h"
#include "operators.h"
#include "relational.h"
#include "inifcns.h"
#include "function.h"
#include "Gt.h"
#include "Gt_helpers.h"
```



## Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::Gt\\_detail](#)

## Functions

- `std::vector< std::vector< int > >` [GiNaC::Gt\\_detail::integer\\_partition](#) (const int n, const int m)
- `template<typename Kernel>`  
[ex GiNaC::Gt\\_detail::deconcatenate\\_path](#) (const std::vector< Kernel > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< Kernel > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `template < ex`  [GiNaC::Gt\\_detail::deconcatenate\\_path< ex >](#) (const std::vector< [ex](#) > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< [ex](#) > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `template < ex`  [GiNaC::Gt\\_detail::deconcatenate\\_path< Gt::kernel >](#) (const std::vector< [Gt::kernel](#) > &args, const std::vector< [ex](#) > &endpoints, const std::function< [ex](#)(std::vector< [Gt::kernel](#) > &new\_args, const [ex](#) &start, const [ex](#) &end)> &construct)
- `std::ostream &` [GiNaC::Gt\\_detail::operator<<](#) (std::ostream &os, const [pathintegral\\_term](#) &term)
- `bool` [GiNaC::Gt\\_detail::operator<](#) (const [pathintegral\\_term](#) &a, const [pathintegral\\_term](#) &b)

### 9.40.1 Detailed Description

Implementation of helper functions for elliptic multiple polylogarithms.

## 9.41 Gt\_helpers.h File Reference

Interface to helper functions for elliptic multiple polylogarithms.

```
#include <set>
#include "ex.h"
#include "hash_map.h"
```

## Classes

- class [GiNaC::Gt\\_detail::TransformExpressionWithCache< Type >](#)
- class [GiNaC::Gt\\_detail::pathintegral\\_term](#)
- class [GiNaC::Gt\\_detail::pathintegral\\_term\\_list](#)

## Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::Gt\\_detail](#)

## Functions

- `std::vector< std::vector< int > > GiNaC::Gt_detail::integer_partition` (const int n, const int m)
- `template<typename Kernel>`  
`ex GiNaC::Gt_detail::deconcatenate_path` (const std::vector< Kernel > &args, const std::vector< ex > &end-points, const std::function< ex(std::vector< Kernel > &new\_args, const ex &start, const ex &end)> &construct)
- `std::ostream & GiNaC::Gt_detail::operator<<` (std::ostream &os, const pathintegral\_term &term)
- `bool GiNaC::Gt_detail::operator<` (const pathintegral\_term &a, const pathintegral\_term &b)

### 9.41.1 Detailed Description

Interface to helper functions for elliptic multiple polylogarithms.

## 9.42 hash\_map.h File Reference

Replacement for `map<>` using hash tables.

```
#include <unordered_map>
```

## Namespaces

- namespace `GiNaC`

## Typedefs

- `template<typename T, class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class Allocator = std::allocator<std::pair<const ex, T>>>`  
`using GiNaC::exhashmap = std::unordered_map<ex, T, Hash, KeyEqual, Allocator>`

### 9.42.1 Detailed Description

Replacement for `map<>` using hash tables.

## 9.43 hash\_seed.h File Reference

Type-specific hash seed.

```
#include <typeinfo>
#include <cstring>
#include "utils.h"
```

## Namespaces

- namespace [GiNaC](#)

### 9.43.1 Detailed Description

Type-specific hash seed.

## 9.44 idx.cpp File Reference

Implementation of [GiNaC](#)'s indices.

```
#include "idx.h"
#include "symbol.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <sstream>
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([idx](#), [basic](#), [print\\_func< print\\_context >\(&idx::do\\_print\)](#), [print\\_func< print\\_latex >\(&idx::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&idx::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&idx::do\\_print\\_tree\)](#)) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([varidx](#))
- [GiNaC::print\\_func< print\\_context > \(&varidx::do\\_print\)](#), [print\\_func< print\\_latex >\(&varidx](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([idx](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([varidx](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([spinidx](#))
- [bool GiNaC::is\\_dummy\\_pair](#) ([const idx &i1](#), [const idx &i2](#))  
*Check whether two indices form a dummy pair.*
- [bool GiNaC::is\\_dummy\\_pair](#) ([const ex &e1](#), [const ex &e2](#))  
*Check whether two expressions form a dummy index pair.*
- [void GiNaC::find\\_free\\_and\\_dummy](#) ([exvector::const\\_iterator it](#), [exvector::const\\_iterator itend](#), [exvector &out\\_free](#), [exvector &out\\_dummy](#))  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- [ex GiNaC::minimal\\_dim](#) ([const ex &dim1](#), [const ex &dim2](#))  
*Return the minimum of two index dimensions.*

## Variables

- [GiNaC::idx](#)

### 9.44.1 Detailed Description

Implementation of [GiNaC](#)'s indices.

## 9.45 idx.h File Reference

Interface to [GiNaC](#)'s indices.

```
#include "ex.h"
#include "numeric.h"
```

## Classes

- class [GiNaC::idx](#)  
*This class holds one index of an indexed object.*
- class [GiNaC::varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [GiNaC::spinidx](#)  
*This class holds a spinor index that can be dotted or undotted and that also has a variance.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([idx](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([varidx](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinidx](#))
- bool [GiNaC::is\\_dummy\\_pair](#) (const [idx](#) &i1, const [idx](#) &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) ([exvector::const\\_iterator](#) it, [exvector::const\\_iterator](#) itend, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (const [exvector](#) &v, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_dummy\\_indices](#) (const [exvector](#) &v, [exvector](#) &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [GiNaC::count\\_dummy\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [GiNaC::count\\_free\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- [ex](#) [GiNaC::minimal\\_dim](#) (const [ex](#) &dim1, const [ex](#) &dim2)  
*Return the minimum of two index dimensions.*

### 9.45.1 Detailed Description

Interface to [GiNaC](#)'s indices.

## 9.46 indexed.cpp File Reference

Implementation of [GiNaC](#)'s indexed expressions.

```
#include "indexed.h"
#include "idx.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "power.h"
#include "relational.h"
#include "symmetry.h"
#include "operators.h"
#include "lst.h"
#include "archive.h"
#include "symbol.h"
#include "utils.h"
#include "integral.h"
#include "matrix.h"
#include "inifcns.h"
#include <algorithm>
#include <iostream>
#include <limits>
#include <sstream>
#include <stdexcept>
```

### Classes

- struct [GiNaC::idx\\_is\\_equal\\_ignore\\_dim](#)
- struct [GiNaC::is\\_summation\\_idx](#)
- struct [GiNaC::ex\\_base\\_is\\_less](#)
- class [GiNaC::terminfo](#)

*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*

- class [GiNaC::terminfo\\_is\\_less](#)
- class [GiNaC::symminfo](#)

*This structure stores the individual symmetrized terms obtained during the simplification of sums.*

- class [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [GiNaC::symminfo\\_is\\_less\\_by\\_orig](#)

### Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([indexed](#), [exprseq](#), [print\\_func](#)< [print\\_context](#) >(&[indexed::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[indexed::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[indexed::do\\_print\\_tree](#))) [indexed](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([indexed](#))
- static bool [GiNaC::indices\\_consistent](#) (const [exvector](#) &v1, const [exvector](#) &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- template<class T>  
[size\\_t GiNaC::number\\_of\\_type](#) (const [exvector](#) &v)
- template<class T>  
[static ex GiNaC::rename\\_dummy\\_indices](#) (const [ex](#) &e, [exvector](#) &global\_dummy\_indices, [exvector](#) &local\_↔  
[\\_dummy\\_indices](#))  
*Rename dummy indices in an expression.*
- static void [GiNaC::find\\_variant\\_indices](#) (const [exvector](#) &v, [exvector](#) &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- bool [GiNaC::reposition\\_dummy\\_indices](#) ([ex](#) &e, [exvector](#) &variant\_dummy\_indices, [exvector](#) &moved\_↔  
indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- static void [GiNaC::product\\_to\\_exvector](#) (const [ex](#) &e, [exvector](#) &v, bool &non\_commutative)
- template<class T>  
[ex GiNaC::idx\\_symmetrization](#) (const [ex](#) &r, const [exvector](#) &local\_dummy\_indices)
- [ex GiNaC::simplify\\_indexed](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const  
[scalar\\_products](#) &sp)  
*Simplify indexed expression, return list of free indices.*
- [ex GiNaC::simplify\\_indexed\\_product](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const  
[scalar\\_products](#) &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free in-  
dices.*
- bool [GiNaC::hasindex](#) (const [ex](#) &x, const [ex](#) &sym)
- [exvector GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const [ex](#) &e)  
*More reliable version of the form.*
- [exvector GiNaC::get\\_all\\_dummy\\_indices](#) (const [ex](#) &e)  
*Returns all dummy indices from the exvector.*
- [lst GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &va, const [exvector](#) &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &va, const [exvector](#) &vb, const [ex](#) &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [ex](#) &a, const [ex](#) &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) ([exvector](#) &va, const [ex](#) &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b  
will be appended to va.*
- [ex GiNaC::expand\\_dummy\\_sum](#) (const [ex](#) &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimen-  
sionality of the dummy index is a nonnegative integer.*

### 9.46.1 Detailed Description

Implementation of [GiNaC](#)'s indexed expressions.

## 9.47 indexed.h File Reference

Interface to [GiNaC](#)'s indexed expressions.

```
#include "exprseq.h"
#include "wildcard.h"
#include <map>
```

### Classes

- class [GiNaC::indexed](#)  
*This class holds an indexed expression.*
- class [GiNaC::spmapkey](#)
- class [GiNaC::scalar\\_products](#)  
*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::map< [spmapkey](#), [ex](#) > [GiNaC::spmap](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([indexed](#))
- [exvector](#) [GiNaC::get\\_all\\_dummy\\_indices](#) (const [ex](#) &[e](#))  
*Returns all dummy indices from the exvector.*
- [exvector](#) [GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const [ex](#) &[e](#))  
*More reliable version of the form.*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) ([exvector](#) &[va](#), const [ex](#) &[b](#), bool [modify\\_va](#)=false)  
*Returns [b](#) with all dummy indices, which are listed in [va](#), renamed if [modify\\_va](#) is set to TRUE all dummy indices of [b](#) will be appended to [va](#).*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [ex](#) &[a](#), const [ex](#) &[b](#))  
*Returns [b](#) with all dummy indices, which are common with [a](#), renamed.*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &[va](#), const [exvector](#) &[vb](#), const [ex](#) &[b](#))  
*Same as above, where [va](#) and [vb](#) contain the indices of [a](#) and [b](#) and are sorted.*
- [lst](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &[va](#), const [exvector](#) &[vb](#))  
*Similar to above, where [va](#) and [vb](#) are the same and the return value is a list of two lists for substitution in [b](#).*
- [ex](#) [GiNaC::expand\\_dummy\\_sum](#) (const [ex](#) &[e](#), bool [subs\\_idx](#)=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

#### 9.47.1 Detailed Description

Interface to [GiNaC](#)'s indexed expressions.

## 9.48 inifcns.cpp File Reference

Implementation of [GiNaC](#)'s initially known functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "lst.h"
#include "fderivative.h"
#include "matrix.h"
#include "mul.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "pseries.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::symbolset](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::conjugate\\_evalf](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_eval](#) (const [ex](#) &arg)
- static void [GiNaC::conjugate\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex](#) [GiNaC::conjugate\\_conjugate](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- static [ex](#) [GiNaC::conjugate\\_real\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_imag\\_part](#) (const [ex](#) &arg)
- static bool [GiNaC::func\\_arg\\_info](#) (const [ex](#) &arg, unsigned inf)
- static bool [GiNaC::conjugate\\_info](#) (const [ex](#) &arg, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (conjugate\_function, eval\_func([conjugate\\_eval](#)). evalf\_func([conjugate\\_evalf](#)).  
expl\_derivative\_func([conjugate\\_expl\\_derivative](#)). info\_func([conjugate\\_info](#)). print\_func<[print\\_latex](#)  
>([conjugate\\_print\\_latex](#)). conjugate\_func([conjugate\\_conjugate](#)). real\_part\_func([conjugate\\_real\\_part](#)).  
imag\_part\_func([conjugate\\_imag\\_part](#)). set\_name("conjugate","conjugate"))
- static [ex](#) [GiNaC::real\\_part\\_evalf](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_eval](#) (const [ex](#) &arg)
- static void [GiNaC::real\\_part\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex](#) [GiNaC::real\\_part\\_conjugate](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_real\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)



- `GiNaC::REGISTER_FUNCTION` (`real_part_function`, `eval_func(real_part_eval)`. `evalf_func(real_part_evalf)`. `expl_derivative_func(real_part_expl_derivative)`. `print_func< print_latex >(real_part_print_latex)`. `conjugate_func(real_part_conjugate)`. `real_part_func(real_part_real_part)`. `imag_part_func(real_part_imag_part)`. `set_name("real_part","real_part")`)
- static `ex` `GiNaC::imag_part_evalf` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_eval` (`const ex &arg`)
- static void `GiNaC::imag_part_print_latex` (`const ex &arg`, `const print_context &c`)
- static `ex` `GiNaC::imag_part_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_real_part` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_expl_derivative` (`const ex &arg`, `const symbol &s`)
- `GiNaC::REGISTER_FUNCTION` (`imag_part_function`, `eval_func(imag_part_eval)`. `evalf_func(imag_part_evalf)`. `expl_derivative_func(imag_part_expl_derivative)`. `print_func< print_latex >(imag_part_print_latex)`. `conjugate_func(imag_part_conjugate)`. `real_part_func(imag_part_real_part)`. `imag_part_func(imag_part_imag_part)`. `set_name("imag_part","imag_part")`)
- static `ex` `GiNaC::abs_evalf` (`const ex &arg`)
- static `ex` `GiNaC::abs_eval` (`const ex &arg`)
- static `ex` `GiNaC::abs_expand` (`const ex &arg`, `unsigned options`)
- static `ex` `GiNaC::abs_expl_derivative` (`const ex &arg`, `const symbol &s`)
- static void `GiNaC::abs_print_latex` (`const ex &arg`, `const print_context &c`)
- static void `GiNaC::abs_print_csrc_float` (`const ex &arg`, `const print_context &c`)
- static `ex` `GiNaC::abs_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::abs_real_part` (`const ex &arg`)
- static `ex` `GiNaC::abs_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::abs_power` (`const ex &arg`, `const ex &exp`)
- bool `GiNaC::abs_info` (`const ex &arg`, `unsigned inf`)
- `GiNaC::REGISTER_FUNCTION` (`abs`, `eval_func(abs_eval)`. `evalf_func(abs_evalf)`. `expand_func(abs_expand)`. `expl_derivative_func(abs_expl_derivative)`. `info_func(abs_info)`. `print_func< print_latex >(abs_print_latex)`. `print_func< print_csrc_float >(abs_print_csrc_float)`. `print_func< print_csrc_double >(abs_print_csrc_double)`. `conjugate_func(abs_conjugate)`. `real_part_func(abs_real_part)`. `imag_part_func(abs_imag_part)`. `power_func(abs_power)`)
- static `ex` `GiNaC::step_evalf` (`const ex &arg`)
- static `ex` `GiNaC::step_eval` (`const ex &arg`)
- static `ex` `GiNaC::step_series` (`const ex &arg`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::step_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::step_real_part` (`const ex &arg`)
- static `ex` `GiNaC::step_imag_part` (`const ex &arg`)
- `GiNaC::REGISTER_FUNCTION` (`step`, `eval_func(step_eval)`. `evalf_func(step_evalf)`. `series_func(step_series)`. `conjugate_func(step_conjugate)`. `real_part_func(step_real_part)`. `imag_part_func(step_imag_part)`)
- static `ex` `GiNaC::csgn_evalf` (`const ex &arg`)
- static `ex` `GiNaC::csgn_eval` (`const ex &arg`)
- static `ex` `GiNaC::csgn_series` (`const ex &arg`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::csgn_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::csgn_real_part` (`const ex &arg`)
- static `ex` `GiNaC::csgn_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::csgn_power` (`const ex &arg`, `const ex &exp`)
- `GiNaC::REGISTER_FUNCTION` (`csgn`, `eval_func(csgn_eval)`. `evalf_func(csgn_evalf)`. `series_func(csgn_series)`. `conjugate_func(csgn_conjugate)`. `real_part_func(csgn_real_part)`. `imag_part_func(csgn_imag_part)`. `power_func(csgn_power)`)
- static `ex` `GiNaC::eta_evalf` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_eval` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_series` (`const ex &x`, `const ex &y`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::eta_conjugate` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_real_part` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_imag_part` (`const ex &x`, `const ex &y`)

- `GiNaC::REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`). `evalf_func(eta_evalf)`. `series_func(eta_series)`. `latex_name("\\eta")`. `set_symmetry(sy_symm(0, 1))`. `conjugate_func(eta_conjugate)`. `real_part_func(eta_real_part)`. `imag_part_func(eta_imag_part)`)
  - static `ex` `GiNaC::Li2_evalf` (const `ex` &`x`)
  - static `ex` `GiNaC::Li2_eval` (const `ex` &`x`)
  - static `ex` `GiNaC::Li2_deriv` (const `ex` &`x`, unsigned `deriv_param`)
  - static `ex` `GiNaC::Li2_series` (const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
  - static `ex` `GiNaC::Li2_conjugate` (const `ex` &`x`)
  - `GiNaC::REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`). `evalf_func(Li2_evalf)`. `derivative_func(Li2_deriv)`. `series_func(Li2_series)`. `conjugate_func(Li2_conjugate)`. `latex_name("\\mathrm{Li}_2")`)
  - static `ex` `GiNaC::Li3_eval` (const `ex` &`x`)
  - `GiNaC::REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`). `latex_name("\\mathrm{Li}_3")`)
  - static `ex` `GiNaC::zetaderiv_eval` (const `ex` &`n`, const `ex` &`x`)
  - static `ex` `GiNaC::zetaderiv_deriv` (const `ex` &`n`, const `ex` &`x`, unsigned `deriv_param`)
  - `GiNaC::REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`). `derivative_func(zetaderiv_deriv)`. `latex_name("\\zeta^{\\prime}")`)
  - static `ex` `GiNaC::factorial_evalf` (const `ex` &`x`)
  - static `ex` `GiNaC::factorial_eval` (const `ex` &`x`)
  - static void `GiNaC::factorial_print_dflt_latex` (const `ex` &`x`, const `print_context` &`c`)
  - static `ex` `GiNaC::factorial_conjugate` (const `ex` &`x`)
  - static `ex` `GiNaC::factorial_real_part` (const `ex` &`x`)
  - static `ex` `GiNaC::factorial_imag_part` (const `ex` &`x`)
  - `GiNaC::REGISTER_FUNCTION` (`factorial`, `eval_func(factorial_eval)`). `evalf_func(factorial_evalf)`. `print_func<print_dflt>(factorial_print_dflt_latex)`. `print_func<print_latex>(factorial_print_dflt_latex)`. `conjugate_func(factorial_conjugate)`. `real_part_func(factorial_real_part)`. `imag_part_func(factorial_imag_part)`)
  - static `ex` `GiNaC::binomial_evalf` (const `ex` &`x`, const `ex` &`y`)
  - static `ex` `GiNaC::binomial_sym` (const `ex` &`x`, const `numeric` &`y`)
  - static `ex` `GiNaC::binomial_eval` (const `ex` &`x`, const `ex` &`y`)
  - static `ex` `GiNaC::binomial_conjugate` (const `ex` &`x`, const `ex` &`y`)
  - static `ex` `GiNaC::binomial_real_part` (const `ex` &`x`, const `ex` &`y`)
  - static `ex` `GiNaC::binomial_imag_part` (const `ex` &`x`, const `ex` &`y`)
  - `GiNaC::REGISTER_FUNCTION` (`binomial`, `eval_func(binomial_eval)`). `evalf_func(binomial_evalf)`. `conjugate_func(binomial_conjugate)`. `real_part_func(binomial_real_part)`. `imag_part_func(binomial_imag_part)`)
  - static `ex` `GiNaC::Order_eval` (const `ex` &`x`)
  - static `ex` `GiNaC::Order_series` (const `ex` &`x`, const `relational` &`r`, int `order`, unsigned `options`)
  - static `ex` `GiNaC::Order_conjugate` (const `ex` &`x`)
  - static `ex` `GiNaC::Order_real_part` (const `ex` &`x`)
  - static `ex` `GiNaC::Order_imag_part` (const `ex` &`x`)
  - static `ex` `GiNaC::Order_power` (const `ex` &`x`, const `ex` &`e`)
  - static `ex` `GiNaC::Order_expl_derivative` (const `ex` &`arg`, const `symbol` &`s`)
  - `GiNaC::REGISTER_FUNCTION` (`Order`, `eval_func(Order_eval)`). `series_func(Order_series)`. `latex_name("\\mathcal{O}")`. `expl_derivative_func(Order_expl_derivative)`. `conjugate_func(Order_conjugate)`. `real_part_func(Order_real_part)`. `imag_part_func(Order_imag_part)`. `power_func(Order_power)`)
  - `ex` `GiNaC::lsolve` (const `ex` &`eqns`, const `ex` &`symbols`, unsigned `options=solve_algo::automatic`)
- Factorial function.*
- const `numeric` `GiNaC::fsolve` (const `ex` &`f`, const `symbol` &`x`, const `numeric` &`x1`, const `numeric` &`x2`)
- Find a real root of real-valued function f(x) numerically within a given interval.*

## Variables

- unsigned `GiNaC::force_include_tgamma` = `tgamma_SERIAL::serial`
- unsigned `GiNaC::force_include_zeta1` = `zeta1_SERIAL::serial`

### 9.48.1 Detailed Description

Implementation of [GiNaC](#)'s initially known functions.

## 9.49 inifcns.h File Reference

Interface to [GiNaC](#)'s initially known functions.

```
#include "numeric.h"
#include "function.h"
#include "ex.h"
```

### Classes

- class [GiNaC::zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [GiNaC::zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*
- class [GiNaC::G2\\_SERIAL](#)  
*Generalized multiple polylogarithm.*
- class [GiNaC::G3\\_SERIAL](#)  
*Generalized multiple polylogarithm with explicit imaginary parts.*
- class [GiNaC::psi1\\_SERIAL](#)  
*Polylogarithm and multiple polylogarithm.*
- class [GiNaC::psi2\\_SERIAL](#)  
*Derivatives of Psi-function (aka polygamma-functions).*
- class [GiNaC::iterated\\_integral2\\_SERIAL](#)  
*Complete elliptic integral of the first kind.*
- class [GiNaC::iterated\\_integral3\\_SERIAL](#)  
*Iterated integral with explicit truncation.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- `template<typename T1>`  
`function GiNaC::zeta (const T1 &p1)`
- `template<typename T1, typename T2>`  
`function GiNaC::zeta (const T1 &p1, const T2 &p2)`
- `template<> bool GiNaC::is\_the\_function< zeta\_SERIAL > (const ex &x)`
- `template<typename T1, typename T2>`  
`function GiNaC::G (const T1 &x, const T2 &y)`
- `template<typename T1, typename T2, typename T3>`  
`function GiNaC::G (const T1 &x, const T2 &s, const T3 &y)`
- `template<> bool GiNaC::is\_the\_function< G\_SERIAL > (const ex &x)`

- `template<typename T1>`  
`function GiNaC::psi` (const T1 &p1)
- `template<typename T1, typename T2>`  
`function GiNaC::psi` (const T1 &p1, const T2 &p2)
- `template<> bool GiNaC::is_the_function< psi_SERIAL >` (const `ex` &x)
- `template<typename T1, typename T2>`  
`function GiNaC::iterated_integral` (const T1 &kernel\_lst, const T2 &lambda)
- `template<typename T1, typename T2, typename T3>`  
`function GiNaC::iterated_integral` (const T1 &kernel\_lst, const T2 &lambda, const T3 &N\_trunc)
- `template<> bool GiNaC::is_the_function< iterated_integral_SERIAL >` (const `ex` &x)
- `ex GiNaC::lsolve` (const `ex` &eqns, const `ex` &symbols, unsigned options=`solve_algo::automatic`)  
*Factorial function.*
- const `numeric GiNaC::fsolve` (const `ex` &f, const `symbol` &x, const `numeric` &x1, const `numeric` &x2)  
*Find a real root of real-valued function f(x) numerically within a given interval.*
- `bool GiNaC::is_order_function` (const `ex` &e)  
*Check whether a function is the Order (O(n)) function.*
- `ex GiNaC::convert_H_to_Li` (const `ex` &parameterlst, const `ex` &arg)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*

### 9.49.1 Detailed Description

Interface to GiNaC's initially known functions.

## 9.50 inifcns\_elliptic.cpp File Reference

Implementation of some special functions related to elliptic curves.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include "integration_kernel.h"
#include "utils_multi_iterator.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- namespace `GiNaC`

## Functions

- static `ex` `GiNaC::EllipticK_evalf` (const `ex` &k)
- static `ex` `GiNaC::EllipticK_eval` (const `ex` &k)
- static `ex` `GiNaC::EllipticK_deriv` (const `ex` &k, unsigned `deriv_param`)
- static `ex` `GiNaC::EllipticK_series` (const `ex` &k, const `relational` &rel, int `order`, unsigned `options`)
- static void `GiNaC::EllipticK_print_latex` (const `ex` &k, const `print_context` &c)
- `GiNaC::REGISTER_FUNCTION` (`EllipticK`, `evalf_func`(`EllipticK_evalf`). `eval_func`(`EllipticK_eval`). `derivative`↔  
`_func`(`EllipticK_deriv`). `series_func`(`EllipticK_series`). `print_func`< `print_latex` >( `EllipticK_print_latex`). `do_`↔  
`not_evalf_params`())
- static `ex` `GiNaC::EllipticE_evalf` (const `ex` &k)
- static `ex` `GiNaC::EllipticE_eval` (const `ex` &k)
- static `ex` `GiNaC::EllipticE_deriv` (const `ex` &k, unsigned `deriv_param`)
- static `ex` `GiNaC::EllipticE_series` (const `ex` &k, const `relational` &rel, int `order`, unsigned `options`)
- static void `GiNaC::EllipticE_print_latex` (const `ex` &k, const `print_context` &c)
- `GiNaC::REGISTER_FUNCTION` (`EllipticE`, `evalf_func`(`EllipticE_evalf`). `eval_func`(`EllipticE_eval`). `derivative`↔  
`_func`(`EllipticE_deriv`). `series_func`(`EllipticE_series`). `print_func`< `print_latex` >( `EllipticE_print_latex`). `do_`↔  
`not_evalf_params`())
- static `ex` `GiNaC::iterated_integral_evalf_impl` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)
- static `ex` `GiNaC::iterated_integral2_evalf` (const `ex` &kernel\_lst, const `ex` &lambda)
- static `ex` `GiNaC::iterated_integral3_evalf` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)
- static `ex` `GiNaC::iterated_integral2_eval` (const `ex` &kernel\_lst, const `ex` &lambda)
- static `ex` `GiNaC::iterated_integral3_eval` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)

### 9.50.1 Detailed Description

Implementation of some special functions related to elliptic curves.

The functions are: complete elliptic integral of the first kind `EllipticK(k)` complete elliptic integral of the second kind `EllipticE(k)` iterated integral `iterated_integral(a,y)` or `iterated_integral(a,y,N_trunc)`

Some remarks:

- All formulae used can be looked up in the following publication: [WW] Numerical evaluation of iterated integrals related to elliptic Feynman integrals, M.Walden, S.Weinzierl, arXiv:2010.05271
- When these routines and methods are used for scientific work that leads to publication in a scientific journal, please refer to this program as : M.Walden, S.Weinzierl, "Numerical evaluation of iterated integrals related to elliptic Feynman integrals", arXiv:2010.05271
- As these routines build on the core part of `GiNaC`, it is also polite to acknowledge C. Bauer, A. Frink, R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language", J. Symbolic Computations 33, 1 (2002), cs.sc/0004015

## 9.51 inifcns\_gamma.cpp File Reference

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

```
#include "inifcns.h"
#include "constant.h"
#include "pseries.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::lgamma\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::lgamma\\_eval](#) (const [ex](#) &x)
 

*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- static [ex](#) [GiNaC::lgamma\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [GiNaC::lgamma\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)
- static [ex](#) [GiNaC::lgamma\\_conjugate](#) (const [ex](#) &x)
- [GiNaC::REGISTER\\_FUNCTION](#) ([lgamma](#), eval\_func([lgamma\\_eval](#)). evalf\_func([lgamma\\_evalf](#)). derivative\_↔\_func([lgamma\\_deriv](#)). series\_func([lgamma\\_series](#)). conjugate\_func([lgamma\\_conjugate](#)). latex\_name("\\log \\Gamma"))
- static [ex](#) [GiNaC::tgamma\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::tgamma\\_eval](#) (const [ex](#) &x)
 

*Evaluation of  $tgamma(x)$ , the true Gamma function.*
- static [ex](#) [GiNaC::tgamma\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [GiNaC::tgamma\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int order, unsigned options)
- static [ex](#) [GiNaC::tgamma\\_conjugate](#) (const [ex](#) &x)
- [GiNaC::REGISTER\\_FUNCTION](#) ([tgamma](#), eval\_func([tgamma\\_eval](#)). evalf\_func([tgamma\\_evalf](#)). derivative\_↔\_func([tgamma\\_deriv](#)). series\_func([tgamma\\_series](#)). conjugate\_func([tgamma\\_conjugate](#)). latex\_↔\_name("\\Gamma"))
- static [ex](#) [GiNaC::beta\\_evalf](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex](#) [GiNaC::beta\\_eval](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex](#) [GiNaC::beta\\_deriv](#) (const [ex](#) &x, const [ex](#) &y, unsigned deriv\_param)
- static [ex](#) [GiNaC::beta\\_series](#) (const [ex](#) &arg1, const [ex](#) &arg2, const [relational](#) &rel, int order, unsigned options)
- [GiNaC::REGISTER\\_FUNCTION](#) ([beta](#), eval\_func([beta\\_eval](#)). evalf\_func([beta\\_evalf](#)). derivative\_↔\_func([beta\\_deriv](#)). series\_func([beta\\_series](#)). latex\_name("\\mathrm{B}"). set\_symmetry([sy\\_symm](#)(0, 1)))
- static [ex](#) [GiNaC::psi1\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::psi1\\_eval](#) (const [ex](#) &x)
 

*Evaluation of digamma-function  $\psi(x)$ .*
- static [ex](#) [GiNaC::psi1\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)

- static `ex` `GiNaC::psi1_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::psi2_evalf` (const `ex` &n, const `ex` &x)
- static `ex` `GiNaC::psi2_eval` (const `ex` &n, const `ex` &x)  
*Evaluation of polygamma-function  $\psi(n,x)$ .*
- static `ex` `GiNaC::psi2_deriv` (const `ex` &n, const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::psi2_series` (const `ex` &n, const `ex` &arg, const `relational` &rel, int order, unsigned options)

### 9.51.1 Detailed Description

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

## 9.52 inifcns\_nstdsums.cpp File Reference

Implementation of some special functions that have a representation as nested sums.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- namespace `GiNaC`

### Functions

- static `ex` `GiNaC::G2_evalf` (const `ex` &x\_, const `ex` &y)
- static `ex` `GiNaC::G2_eval` (const `ex` &x\_, const `ex` &y)
- static `ex` `GiNaC::G3_evalf` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex` `GiNaC::G3_eval` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex` `GiNaC::Li_evalf` (const `ex` &m\_, const `ex` &x\_)
- static `ex` `GiNaC::Li_eval` (const `ex` &m\_, const `ex` &x\_)
- static `ex` `GiNaC::Li_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::Li_deriv` (const `ex` &m\_, const `ex` &x\_, unsigned deriv\_param)
- static void `GiNaC::Li_print_latex` (const `ex` &m\_, const `ex` &x\_, const `print_context` &c)

- `GiNaC::REGISTER_FUNCTION` (`Li`, `evalf_func(Li_evalf)`. `eval_func(Li_eval)`. `series_func(Li_series)`. `derivative_func(Li_deriv)`. `print_func< print_latex >(Li_print_latex)`. `do_not_evalf_params()`)
- static `ex` `GiNaC::S_evalf` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex` `GiNaC::S_eval` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex` `GiNaC::S_series` (const `ex` &n, const `ex` &p, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::S_deriv` (const `ex` &n, const `ex` &p, const `ex` &x, unsigned deriv\_param)
- static void `GiNaC::S_print_latex` (const `ex` &n, const `ex` &p, const `ex` &x, const `print_context` &c)
- `GiNaC::REGISTER_FUNCTION` (`S`, `evalf_func(S_evalf)`. `eval_func(S_eval)`. `series_func(S_series)`. `derivative_func(S_deriv)`. `print_func< print_latex >(S_print_latex)`. `do_not_evalf_params()`)
- static `ex` `GiNaC::H_evalf` (const `ex` &x1, const `ex` &x2)
- static `ex` `GiNaC::H_eval` (const `ex` &m\_, const `ex` &x)
- static `ex` `GiNaC::H_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::H_deriv` (const `ex` &m\_, const `ex` &x, unsigned deriv\_param)
- static void `GiNaC::H_print_latex` (const `ex` &m\_, const `ex` &x, const `print_context` &c)
- `GiNaC::REGISTER_FUNCTION` (`H`, `evalf_func(H_evalf)`. `eval_func(H_eval)`. `series_func(H_series)`. `derivative_func(H_deriv)`. `print_func< print_latex >(H_print_latex)`. `do_not_evalf_params()`)
- `ex` `GiNaC::convert_H_to_Li` (const `ex` &parameterlst, const `ex` &arg)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*
- static `ex` `GiNaC::zeta1_evalf` (const `ex` &x)
- static `ex` `GiNaC::zeta1_eval` (const `ex` &m)
- static `ex` `GiNaC::zeta1_deriv` (const `ex` &m, unsigned deriv\_param)
- static void `GiNaC::zeta1_print_latex` (const `ex` &m\_, const `print_context` &c)
- static `ex` `GiNaC::zeta2_evalf` (const `ex` &x, const `ex` &s)
- static `ex` `GiNaC::zeta2_eval` (const `ex` &m, const `ex` &s\_)
- static `ex` `GiNaC::zeta2_deriv` (const `ex` &m, const `ex` &s, unsigned deriv\_param)
- static void `GiNaC::zeta2_print_latex` (const `ex` &m\_, const `ex` &s\_, const `print_context` &c)

### 9.52.1 Detailed Description

Implementation of some special functions that have a representation as nested sums.

The functions are: classical polylogarithm  $\text{Li}(n,x)$  multiple polylogarithm  $\text{Li}(\{m_1,\dots,m_k\},\{x_1,\dots,x_k\})$   $G(\{a_1,\dots,a_k\},y)$  or  $G(\{a_1,\dots,a_k\},\{s_1,\dots,s_k\},y)$  Nielsen's generalized polylogarithm  $S(n,p,x)$  harmonic polylogarithm  $H(m,x)$  or  $H(\{m_1,\dots,m_k\},x)$  multiple zeta value  $\zeta(m)$  or  $\zeta(\{m_1,\dots,m_k\})$  alternating Euler sum  $\zeta(m,s)$  or  $\zeta(\{m_1,\dots,m_k\},\{s_1,\dots,s_k\})$

Some remarks:

- All formulae used can be looked up in the following publications: [Kol] Nielsen's Generalized Polylogarithms, K.S.Kolbig, SIAM J.Math.Anal. 17 (1986), pp. 1232-1258. [Cra] Fast Evaluation of Multiple Zeta Sums, R.E.Crandall, Math.Comp. 67 (1998), pp. 1163-1172. [ReV] Harmonic Polylogarithms, E.Remiddi, J.A. Vermaseren, Int.J.Mod.Phys. A15 (2000), pp. 725-754 [BBB] Special Values of Multiple Polylogarithms, J.Borwein, D.Bradley, D.Broadhurst, P.Lisonek, Trans.Amer.Math.Soc. 353/3 (2001), pp. 907-941 [VSW] Numerical evaluation of multiple polylogarithms, J.Vollinga, S.Weinzierl, hep-ph/0410259
- The order of parameters and arguments of Li and zeta is defined according to the nested sums representation. The parameters for H are understood as in [ReV]. They can be in expanded — only 0, 1 and -1 — or in compactified — a string with zeros in front of 1 or -1 is written as a single number — notation.
- All functions can be numerically evaluated with arguments in the whole complex plane. The parameters for Li, zeta and S must be positive integers. If you want to have an alternating Euler sum, you have to give the signs of the parameters as a second argument s to  $\zeta(m,s)$  containing 1 and -1.



- The calculation of classical polylogarithms is speeded up by using Bernoulli numbers and look-up tables. S uses look-up tables as well. The zeta function applies the algorithms in [Cra] and [BBB] for speed up. Multiple polylogarithms use Hoelder convolution [BBB].
- The functions have no means to do a series expansion into nested sums. To do this, you have to convert these functions into the appropriate objects from the nestedsums library, do the expansion and convert the result back.
- Numerical testing of this implementation has been performed by doing a comparison of results between this software and the commercial M..... 4.1. Multiple zeta values have been checked by means of evaluations into simple zeta values. Harmonic polylogarithms have been checked by comparison to  $S(n,p,x)$  for corresponding parameter combinations and by continuity checks around  $|x|=1$  along with comparisons to corresponding zeta functions. Multiple polylogarithms were checked against H and zeta and by means of shuffle and quasi-shuffle relations.

## 9.53 inifcns\_trans.cpp File Reference

Implementation of transcendental (and trigonometric and hyperbolic) functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "add.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include "pseries.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::exp\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_eval](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_expand](#) (const [ex](#) &arg, unsigned options)
- static [ex](#) [GiNaC::exp\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [GiNaC::exp\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_conjugate](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_power](#) (const [ex](#) &x, const [ex](#) &a)
- static bool [GiNaC::exp\\_info](#) (const [ex](#) &x, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) ([exp](#), [eval\\_func](#)([exp\\_eval](#)). [evalf\\_func](#)([exp\\_evalf](#)). [info\\_func](#)([exp\\_info](#)). [expand\\_func](#)([exp\\_expand](#)). [derivative\\_func](#)([exp\\_deriv](#)). [real\\_part\\_func](#)([exp\\_real\\_part](#)). [imag\\_part\\_func](#)([exp\\_imag\\_part](#)). [conjugate\\_func](#)([exp\\_conjugate](#)). [power\\_func](#)([exp\\_power](#)). [latex\\_name](#)("\\exp"))

- static `ex` `GiNaC::log_evalf` (const `ex` &x)
- static `ex` `GiNaC::log_eval` (const `ex` &x)
- static `ex` `GiNaC::log_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::log_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::log_real_part` (const `ex` &x)
- static `ex` `GiNaC::log_imag_part` (const `ex` &x)
- static `ex` `GiNaC::log_expand` (const `ex` &arg, unsigned options)
- static `ex` `GiNaC::log_conjugate` (const `ex` &x)
- static bool `GiNaC::log_info` (const `ex` &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (`log`, eval\_func(`log_eval`), evalf\_func(`log_evalf`), info\_func(`log_info`), expand\_func(`log_expand`), derivative\_func(`log_deriv`), series\_func(`log_series`), real\_part\_func(`log_real_part`), imag\_part\_func(`log_imag_part`), conjugate\_func(`log_conjugate`), latex\_name("\\ln"))
- static `ex` `GiNaC::sin_evalf` (const `ex` &x)
- static `ex` `GiNaC::sin_eval` (const `ex` &x)
- static `ex` `GiNaC::sin_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::sin_real_part` (const `ex` &x)
- static `ex` `GiNaC::sin_imag_part` (const `ex` &x)
- static `ex` `GiNaC::sin_conjugate` (const `ex` &x)
- static bool `GiNaC::trig_info` (const `ex` &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (`sin`, eval\_func(`sin_eval`), evalf\_func(`sin_evalf`), info\_func(`trig_info`), derivative\_func(`sin_deriv`), real\_part\_func(`sin_real_part`), imag\_part\_func(`sin_imag_part`), conjugate\_func(`sin_conjugate`), latex\_name("\\sin"))
- static `ex` `GiNaC::cos_evalf` (const `ex` &x)
- static `ex` `GiNaC::cos_eval` (const `ex` &x)
- static `ex` `GiNaC::cos_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::cos_real_part` (const `ex` &x)
- static `ex` `GiNaC::cos_imag_part` (const `ex` &x)
- static `ex` `GiNaC::cos_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`cos`, eval\_func(`cos_eval`), info\_func(`trig_info`), evalf\_func(`cos_evalf`), derivative\_func(`cos_deriv`), real\_part\_func(`cos_real_part`), imag\_part\_func(`cos_imag_part`), conjugate\_func(`cos_conjugate`), latex\_name("\\cos"))
- static `ex` `GiNaC::tan_evalf` (const `ex` &x)
- static `ex` `GiNaC::tan_eval` (const `ex` &x)
- static `ex` `GiNaC::tan_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::tan_real_part` (const `ex` &x)
- static `ex` `GiNaC::tan_imag_part` (const `ex` &x)
- static `ex` `GiNaC::tan_series` (const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::tan_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`tan`, eval\_func(`tan_eval`), evalf\_func(`tan_evalf`), info\_func(`trig_info`), derivative\_func(`tan_deriv`), series\_func(`tan_series`), real\_part\_func(`tan_real_part`), imag\_part\_func(`tan_imag_part`), conjugate\_func(`tan_conjugate`), latex\_name("\\tan"))
- static `ex` `GiNaC::asin_evalf` (const `ex` &x)
- static `ex` `GiNaC::asin_eval` (const `ex` &x)
- static `ex` `GiNaC::asin_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::asin_conjugate` (const `ex` &x)
- static bool `GiNaC::asin_info` (const `ex` &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (`asin`, eval\_func(`asin_eval`), evalf\_func(`asin_evalf`), info\_func(`asin_info`), derivative\_func(`asin_deriv`), conjugate\_func(`asin_conjugate`), latex\_name("\\arcsin"))
- static `ex` `GiNaC::acos_evalf` (const `ex` &x)
- static `ex` `GiNaC::acos_eval` (const `ex` &x)
- static `ex` `GiNaC::acos_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::acos_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`acos`, eval\_func(`acos_eval`), evalf\_func(`acos_evalf`), info\_func(`asin_info`), derivative\_func(`acos_deriv`), conjugate\_func(`acos_conjugate`), latex\_name("\\arccos"))
- static `ex` `GiNaC::atan_evalf` (const `ex` &x)

- static `ex` `GiNaC::atan_eval` (const `ex` &x)
- static `ex` `GiNaC::atan_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::atan_series` (const `ex` &arg, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::atan_conjugate` (const `ex` &x)
- static bool `GiNaC::atan_info` (const `ex` &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (`atan`, `eval_func(atan_eval)`. `evalf_func(atan_evalf)`. `info_func(atan_info)`. `derivative_func(atan_deriv)`. `series_func(atan_series)`. `conjugate_func(atan_conjugate)`. `latex_name("\\arctan")`)
- static `ex` `GiNaC::atan2_evalf` (const `ex` &y, const `ex` &x)
- static `ex` `GiNaC::atan2_eval` (const `ex` &y, const `ex` &x)
- static `ex` `GiNaC::atan2_deriv` (const `ex` &y, const `ex` &x, unsigned deriv\_param)
- static bool `GiNaC::atan2_info` (const `ex` &y, const `ex` &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (`atan2`, `eval_func(atan2_eval)`. `evalf_func(atan2_evalf)`. `info_func(atan2_info)`. `evalf_func(atan2_evalf)`. `derivative_func(atan2_deriv)`)
- static `ex` `GiNaC::sinh_evalf` (const `ex` &x)
- static `ex` `GiNaC::sinh_eval` (const `ex` &x)
- static `ex` `GiNaC::sinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::sinh_real_part` (const `ex` &x)
- static `ex` `GiNaC::sinh_imag_part` (const `ex` &x)
- static `ex` `GiNaC::sinh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`sinh`, `eval_func(sinh_eval)`. `evalf_func(sinh_evalf)`. `info_func(atan_info)`. `derivative_func(sinh_deriv)`. `real_part_func(sinh_real_part)`. `imag_part_func(sinh_imag_part)`. `conjugate_func(sinh_conjugate)`. `latex_name("\\sinh")`)
- static `ex` `GiNaC::cosh_evalf` (const `ex` &x)
- static `ex` `GiNaC::cosh_eval` (const `ex` &x)
- static `ex` `GiNaC::cosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::cosh_real_part` (const `ex` &x)
- static `ex` `GiNaC::cosh_imag_part` (const `ex` &x)
- static `ex` `GiNaC::cosh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`cosh`, `eval_func(cosh_eval)`. `evalf_func(cosh_evalf)`. `info_func(exp_info)`. `derivative_func(cosh_deriv)`. `real_part_func(cosh_real_part)`. `imag_part_func(cosh_imag_part)`. `conjugate_func(cosh_conjugate)`. `latex_name("\\cosh")`)
- static `ex` `GiNaC::tanh_evalf` (const `ex` &x)
- static `ex` `GiNaC::tanh_eval` (const `ex` &x)
- static `ex` `GiNaC::tanh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::tanh_series` (const `ex` &x, const `relational` &rel, int order, unsigned options)
- static `ex` `GiNaC::tanh_real_part` (const `ex` &x)
- static `ex` `GiNaC::tanh_imag_part` (const `ex` &x)
- static `ex` `GiNaC::tanh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`tanh`, `eval_func(tanh_eval)`. `evalf_func(tanh_evalf)`. `info_func(atan_info)`. `derivative_func(tanh_deriv)`. `series_func(tanh_series)`. `real_part_func(tanh_real_part)`. `imag_part_func(tanh_imag_part)`. `conjugate_func(tanh_conjugate)`. `latex_name("\\tanh")`)
- static `ex` `GiNaC::asinh_evalf` (const `ex` &x)
- static `ex` `GiNaC::asinh_eval` (const `ex` &x)
- static `ex` `GiNaC::asinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::asinh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`asinh`, `eval_func(asinh_eval)`. `evalf_func(asinh_evalf)`. `info_func(atan_info)`. `derivative_func(asinh_deriv)`. `conjugate_func(asinh_conjugate)`)
- static `ex` `GiNaC::acosh_evalf` (const `ex` &x)
- static `ex` `GiNaC::acosh_eval` (const `ex` &x)
- static `ex` `GiNaC::acosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex` `GiNaC::acosh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`acosh`, `eval_func(acosh_eval)`. `evalf_func(acosh_evalf)`. `info_func(asin_info)`. `derivative_func(acosh_deriv)`. `conjugate_func(acosh_conjugate)`)
- static `ex` `GiNaC::atanh_evalf` (const `ex` &x)

- static `ex` `GiNaC::atanh_eval` (const `ex` &x)
- static `ex` `GiNaC::atanh_deriv` (const `ex` &x, unsigned `deriv_param`)
- static `ex` `GiNaC::atanh_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex` `GiNaC::atanh_conjugate` (const `ex` &x)
- `GiNaC::REGISTER_FUNCTION` (`atanh`, `eval_func(atanh_eval)`. `evalf_func(atanh_evalf)`. `info_↔`  
`func(asin_info)`. `derivative_func(atanh_deriv)`. `series_func(atanh_series)`. `conjugate_func(atanh_conjugate)`)

### 9.53.1 Detailed Description

Implementation of transcendental (and trigonometric and hyperbolic) functions.

## 9.54 `integral.cpp` File Reference

Implementation of `GiNaC`'s symbolic integral.

```
#include "integral.h"
#include "numeric.h"
#include "symbol.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "inifcns.h"
#include "wildcard.h"
#include "archive.h"
#include "registrar.h"
#include "utils.h"
#include "operators.h"
#include "relational.h"
```

### Classes

- struct `GiNaC::error_and_integral`
- struct `GiNaC::error_and_integral_is_less`

### Namespaces

- namespace `GiNaC`

### Typedefs

- typedef `map< error_and_integral, ex, error_and_integral_is_less >` `GiNaC::lookup_map`

### Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integral`, `basic`, `print_func< print_dflt >(&integral::do_print)`. `print_func< print_python >(&integral::do_print)`. `print_func< print_latex >(&integral::do_print_latex)`) `integral`
- `ex` `GiNaC::subvalue` (const `ex` &var, const `ex` &value, const `ex` &fun)
- `ex` `GiNaC::adaptivesimpson` (const `ex` &x, const `ex` &a\_in, const `ex` &b\_in, const `ex` &f, const `ex` &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- `GiNaC::GINAC_BIND_UNARCHIVER` (`integral`)

### 9.54.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic integral.

## 9.55 integral.h File Reference

Interface to [GiNaC](#)'s symbolic integral.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::integral](#)  
*Symbolic integral.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([integral](#))
- [ex GiNaC::adaptivesimpson](#) (const [ex](#) &x, const [ex](#) &a\_in, const [ex](#) &b\_in, const [ex](#) &f, const [ex](#) &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*

### 9.55.1 Detailed Description

Interface to [GiNaC](#)'s symbolic integral.

## 9.56 integration\_kernel.cpp File Reference

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "integration_kernel.h"
#include "add.h"
#include "mul.h"
#include "operators.h"
#include "power.h"
#include "relational.h"
#include "symbol.h"
#include "constant.h"
#include "numeric.h"
#include "function.h"
#include "pseries.h"
#include "utils.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <cln/cln.h>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [ex GiNaC::ifactor](#) (const [numeric](#) &n)  
*Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst(p_1, \dots, pr), lst(a_1, \dots, ar)$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .*
- [bool GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const [numeric](#) &n)  
*Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.*
- [numeric GiNaC::kronecker\\_symbol](#) (const [numeric](#) &a, const [numeric](#) &n)  
*Returns the Kronecker symbol  $a: integer\ n: integer$ .*
- [numeric GiNaC::primitive\\_dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- [numeric GiNaC::dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a, const [numeric](#) &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- [numeric GiNaC::generalised\\_Bernoulli\\_number](#) (const [numeric](#) &k, const [numeric](#) &b)  
*The generalised Bernoulli number.*
- [ex GiNaC::Bernoulli\\_polynomial](#) (const [numeric](#) &k, const [ex](#) &x)  
*The Bernoulli polynomials.*
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< print\_context >(&integration\_kernel::do\_print)) integration\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_func< print\_context >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< print\_context >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< print\_context >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< print\_context >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< print\_context >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< print\_context >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)

### 9.56.1 Detailed Description

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

## 9.57 integration\_kernel.h File Reference

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "basic.h"
#include "archive.h"
#include "numeric.h"
#include "lst.h"
#include <cln/complex.h>
#include <vector>
```

### Classes

- class [GiNaC::integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- class [GiNaC::basic\\_log\\_kernel](#)  
The basic integration kernel with a logarithmic singularity at the origin.
- class [GiNaC::multiple\\_polylog\\_kernel](#)  
The integration kernel for multiple polylogarithms.
- class [GiNaC::ELi\\_kernel](#)  
The ELi-kernel.
- class [GiNaC::Ebar\\_kernel](#)  
The Ebar-kernel.
- class [GiNaC::Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [GiNaC::Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [GiNaC::Eisenstein\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .
- class [GiNaC::Eisenstein\\_h\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .
- class [GiNaC::modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [GiNaC::user\\_defined\\_kernel](#)  
A user-defined integration kernel.

### Namespaces

- namespace [GiNaC](#)

## Functions

- `ex GiNaC::ifactor` (const `numeric` &n)  
Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst(lst(p_1, \dots, pr), lst(a_1, \dots, ar))$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .
- `bool GiNaC::is_discriminant_of_quadratic_number_field` (const `numeric` &n)  
Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.
- `numeric GiNaC::kronecker_symbol` (const `numeric` &a, const `numeric` &n)  
Returns the Kronecker symbol  $a: integer\ n: integer$ .
- `numeric GiNaC::primitive_dirichlet_character` (const `numeric` &n, const `numeric` &a)  
Defines a primitive Dirichlet character through the Kronecker symbol.
- `numeric GiNaC::dirichlet_character` (const `numeric` &n, const `numeric` &a, const `numeric` &N)  
Defines a Dirichlet character through the Kronecker symbol.
- `numeric GiNaC::generalised_Bernoulli_number` (const `numeric` &k, const `numeric` &b)  
The generalised Bernoulli number.
- `ex GiNaC::Bernoulli_polynomial` (const `numeric` &k, const `ex` &x)  
The Bernoulli polynomials.
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (integration\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (basic\_log\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (multiple\_polylog\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (ELi\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (Ebar\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (Kronecker\_dtau\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (Kronecker\_dz\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (Eisenstein\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (Eisenstein\_h\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (modular\_form\_kernel)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (user\_defined\_kernel)

### 9.57.1 Detailed Description

Interface to `GiNaC`'s integration kernels for iterated integrals.

## 9.58 Ist.cpp File Reference

Implementation of `GiNaC`'s `Ist`.

```
#include "lst.h"
#include "archive.h"
```

## Namespaces

- namespace `GiNaC`

## Variables

- `template<> GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(Ist, basic, print_func< print_context >(&Ist::do_print). print_func< print_tree >(&Ist::do_print_tree))` `template<> bool Is GiNaC::GINAC_BIND_UNARCHIVER(Ist)`  
Specialization of `container::info()` for `Ist`.



### 9.58.1 Detailed Description

Implementation of [GiNaC](#)'s Ist.

## 9.59 Ist.h File Reference

Definition of [GiNaC](#)'s Ist.

```
#include "container.h"  
#include <list>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef [container](#)< std::list > [GiNaC::Ist](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Ist)

### 9.59.1 Detailed Description

Definition of [GiNaC](#)'s Ist.

## 9.60 matrix.cpp File Reference

Implementation of symbolic matrices.

```
#include "matrix.h"  
#include "numeric.h"  
#include "lst.h"  
#include "idx.h"  
#include "indexed.h"  
#include "add.h"  
#include "power.h"  
#include "symbol.h"  
#include "operators.h"  
#include "normal.h"  
#include "archive.h"  
#include "utils.h"  
#include <map>  
#include <sstream>  
#include <stdexcept>  
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([matrix](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[matrix::do\\_print](#)). [print\\_func](#)< [print\\_latex](#) >(&[matrix::do\\_print\\_latex](#)). [print\\_func](#)< [print\\_tree](#) >(&[matrix::do\\_print\\_tree](#)). [print\\_func](#)< [print\\_python\\_repr](#) >(&[matrix::do\\_print\\_python\\_repr](#))) [matrix](#)  
*Default ctor.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([matrix](#))
- [ex GiNaC::lst\\_to\\_matrix](#) (const [lst](#) &l)  
*Convert list of lists to matrix.*
- [ex GiNaC::diag\\_matrix](#) (const [lst](#) &l)  
*Convert list of diagonal elements to matrix.*
- [ex GiNaC::diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex GiNaC::unit\\_matrix](#) (unsigned r, unsigned c)  
*Create an r times c unit matrix.*
- [ex GiNaC::symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↵  
base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex GiNaC::reduced\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned c)  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex GiNaC::sub\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned nr, unsigned c, unsigned nc)  
*Return the nr times nc submatrix starting at position r, c of matrix m.*

### 9.60.1 Detailed Description

Implementation of symbolic matrices.

## 9.61 matrix.h File Reference

Interface to symbolic matrices.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include "compiler.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::matrix](#)  
*Symbolic matrices.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t GiNaC::nops](#) (const [matrix](#) &m)
- [ex GiNaC::expand](#) (const [matrix](#) &m, unsigned options=0)
- [ex GiNaC::evalf](#) (const [matrix](#) &m)
- unsigned [GiNaC::rows](#) (const [matrix](#) &m)
- unsigned [GiNaC::cols](#) (const [matrix](#) &m)
- [matrix GiNaC::transpose](#) (const [matrix](#) &m)
- [ex GiNaC::determinant](#) (const [matrix](#) &m, unsigned options=[determinant\\_algo::automatic](#))
- [ex GiNaC::trace](#) (const [matrix](#) &m)
- [ex GiNaC::charpoly](#) (const [matrix](#) &m, const [ex](#) &lambda)
- [matrix GiNaC::inverse](#) (const [matrix](#) &m)
- [matrix GiNaC::inverse](#) (const [matrix](#) &m, unsigned algo)
- unsigned [GiNaC::rank](#) (const [matrix](#) &m)
- unsigned [GiNaC::rank](#) (const [matrix](#) &m, unsigned [solve\\_algo](#))
- [ex GiNaC::lst\\_to\\_matrix](#) (const [lst](#) &l)  
*Convert list of lists to matrix.*
- [ex GiNaC::diag\\_matrix](#) (const [lst](#) &l)  
*Convert list of diagonal elements to matrix.*
- [ex GiNaC::diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex GiNaC::unit\\_matrix](#) (unsigned r, unsigned c)  
*Create an r times c unit matrix.*
- [ex GiNaC::unit\\_matrix](#) (unsigned x)  
*Create a x times x unit matrix.*
- [ex GiNaC::symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↵  
base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex GiNaC::reduced\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned c)  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex GiNaC::sub\\_matrix](#) (const [matrix](#) &m, unsigned r, unsigned nr, unsigned c, unsigned nc)  
*Return the nr times nc submatrix starting at position r, c of matrix m.*
- [ex GiNaC::symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*

### 9.61.1 Detailed Description

Interface to symbolic matrices.

## 9.62 mul.cpp File Reference

Implementation of [GiNaC](#)'s products of expressions.

```
#include "mul.h"
#include "add.h"
#include "power.h"
#include "operators.h"
#include "matrix.h"
#include "indexed.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "symbol.h"
#include "compiler.h"
#include <limits>
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([mul](#), [expairseq](#), [print\\_func< print\\_context >\(&mul::do\\_print\)](#), [print\\_func< print\\_latex >\(&mul::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&mul::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&mul::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&mul::do\\_print\\_python\\_repr\)](#))  
[mul](#)
- [bool GiNaC::tryfactsubs](#) ([const ex &origfactor](#), [const ex &patternfactor](#), [int &nummatches](#), [exmap &repls](#))
- [bool GiNaC::algebraic\\_match\\_mul\\_with\\_mul](#) ([const mul &e](#), [const ex &pat](#), [exmap &repls](#), [int factor](#), [int &nummatches](#), [const std::vector< bool > &subsed](#), [std::vector< bool > &matched](#))  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([mul](#))

### 9.62.1 Detailed Description

Implementation of [GiNaC](#)'s products of expressions.

## 9.63 mul.h File Reference

Interface to [GiNaC](#)'s products of expressions.

```
#include "expairseq.h"
```

## Classes

- class [GiNaC::mul](#)  
*Product of expressions.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([mul](#))

### 9.63.1 Detailed Description

Interface to [GiNaC](#)'s products of expressions.

## 9.64 ncmul.cpp File Reference

Implementation of [GiNaC](#)'s non-commutative products of expressions.

```
#include "ncmul.h"  
#include "ex.h"  
#include "add.h"  
#include "mul.h"  
#include "clifford.h"  
#include "matrix.h"  
#include "archive.h"  
#include "indexed.h"  
#include "utils.h"  
#include <algorithm>  
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< std::size\_t > [GiNaC::uintvector](#)
- typedef std::vector< unsigned > [GiNaC::unsignedvector](#)
- typedef std::vector< [exvector](#) > [GiNaC::exvectorvector](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([ncmul](#), [exprseq](#), [print\\_func](#)< [print\\_context](#) >(&[ncmul::do\\_print](#)), [print\\_func](#)< [print\\_tree](#) >(&[ncmul::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_csrc](#) >(&[ncmul::do\\_print\\_csrc](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[ncmul::do\\_print\\_csrc](#))) [ncmul](#)
- [ex GiNaC::reeval\\_ncmul](#) (const [exvector](#) &[v](#))
- [ex GiNaC::hold\\_ncmul](#) (const [exvector](#) &[v](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([ncmul](#))

### 9.64.1 Detailed Description

Implementation of [GiNaC](#)'s non-commutative products of expressions.

## 9.65 ncmul.h File Reference

Interface to [GiNaC](#)'s non-commutative products of expressions.

```
#include "exprseq.h"
#include "archive.h"
```

## Classes

- class [GiNaC::ncmul](#)  
*Non-commutative product of expressions.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([ncmul](#))
- [ex GiNaC::reeval\\_ncmul](#) (const [exvector](#) &[v](#))
- [ex GiNaC::hold\\_ncmul](#) (const [exvector](#) &[v](#))

### 9.65.1 Detailed Description

Interface to [GiNaC](#)'s non-commutative products of expressions.

## 9.66 normal.cpp File Reference

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "normal.h"
#include "basic.h"
#include "ex.h"
#include "add.h"
#include "constant.h"
#include "expairseq.h"
#include "fail.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "matrix.h"
#include "pseries.h"
#include "symbol.h"
#include "utils.h"
#include "polynomial/chinrem_gcd.h"
#include <algorithm>
#include <map>
```

### Classes

- struct [GiNaC::sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [GiNaC::gcdheu\\_failed](#)  
*Exception thrown by [heur\\_gcd\(\)](#) to signal failure.*
- struct [GiNaC::normal\\_map\\_function](#)  
*Function object to be applied by [basic::normal\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

### Macros

- `#define` [FAST\\_COMPARE](#) 1
- `#define` [USE\\_REMEMBER](#) 0
- `#define` [USE\\_TRIAL\\_DIVISION](#) 0
- `#define` [STATISTICS](#) 0

### Typedefs

- `typedef` `std::vector< sym\_desc >` [GiNaC::sym\\_desc\\_vec](#)

## Functions

- static bool `GiNaC::get_first_symbol` (const `ex` &e, `ex` &x)  
*Return pointer to first symbol found in expression.*
- static void `GiNaC::add_symbol` (const `ex` &s, `sym_desc_vec` &v)
- static void `GiNaC::collect_symbols` (const `ex` &e, `sym_desc_vec` &v)
- static void `GiNaC::get_symbol_stats` (const `ex` &a, const `ex` &b, `sym_desc_vec` &v)  
*Collect statistical information about symbols in polynomials.*
- static `numeric` `GiNaC::lcmcoeff` (const `ex` &e, const `numeric` &l)
- static `numeric` `GiNaC::lcm_of_coefficients_denominators` (const `ex` &e)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static `ex` `GiNaC::multiply_lcm` (const `ex` &e, const `numeric` &lcm)  
*Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.*
- `ex` `GiNaC::quo` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- `ex` `GiNaC::rem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- `ex` `GiNaC::decomp_rational` (const `ex` &a, const `ex` &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- `ex` `GiNaC::prem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- `ex` `GiNaC::sprem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool `GiNaC::divide` (const `ex` &a, const `ex` &b, `ex` &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
- static bool `GiNaC::divide_in_z` (const `ex` &a, const `ex` &b, `ex` &q, `sym_desc_vec::const_iterator` var)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .*
- static `ex` `GiNaC::sr_gcd` (const `ex` &a, const `ex` &b, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
- static `ex` `GiNaC::interpolate` (const `ex` &gamma, const `numeric` &xi, const `ex` &x, int degree\_hint=1)  
 *$\xi$ -adic polynomial interpolation*
- static bool `GiNaC::heur_gcd_z` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static bool `GiNaC::heur_gcd` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static `ex` `GiNaC::gcd_pf_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- static `ex` `GiNaC::gcd_pf_mul` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex` `GiNaC::gcd` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, bool check\_args, unsigned options)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- static `ex` `GiNaC::gcd_pf_pow_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex` `GiNaC::lcm` (const `ex` &a, const `ex` &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- static `epvector` `GiNaC::sqrfree_yun` (const `ex` &a, const `symbol` &x)  
*Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
- `ex` `GiNaC::sqrfree` (const `ex` &a, const `lst` &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- `ex` `GiNaC::sqrfree_parfrac` (const `ex` &a, const `symbol` &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- static `ex` `GiNaC::replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*



- static `ex GiNaC::replace_with_symbol` (const `ex` &e, `exmap` &repl)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex GiNaC::frac_cancel` (const `ex` &n, const `ex` &d)  
*Fraction cancellation.*
- static `ex GiNaC::find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)  
*Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
- `ex GiNaC::collect_common_factors` (const `ex` &e)  
*Collect common factors in sums.*
- `ex GiNaC::resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)  
*Resultant of two expressions e1,e2 with respect to symbol s.*

### 9.66.1 Detailed Description

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

### 9.66.2 Macro Definition Documentation

#### 9.66.2.1 FAST\_COMPARE

```
#define FAST_COMPARE 1
```

#### 9.66.2.2 USE\_REMEMBER

```
#define USE_REMEMBER 0
```

#### 9.66.2.3 USE\_TRIAL\_DIVISION

```
#define USE_TRIAL_DIVISION 0
```

#### 9.66.2.4 STATISTICS

```
#define STATISTICS 0
```

## 9.67 normal.h File Reference

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "lst.h"
```

## Classes

- struct [GiNaC::gcd\\_options](#)  
*Flags to control the behavior of [gcd\(\)](#) and friends.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [ex GiNaC::quo](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::rem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::decomp\\_rational](#) (const [ex](#) &a, const [ex](#) &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- [ex GiNaC::prem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::sprem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [bool GiNaC::divide](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
- [ex GiNaC::gcd](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) \*ca, [ex](#) \*cb, bool check\_args, unsigned options)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- [ex GiNaC::lcm](#) (const [ex](#) &a, const [ex](#) &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- [ex GiNaC::sqrfree](#) (const [ex](#) &a, const [lst](#) &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- [ex GiNaC::sqrfree\\_parfrac](#) (const [ex](#) &a, const [symbol](#) &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- [ex GiNaC::collect\\_common\\_factors](#) (const [ex](#) &e)  
*Collect common factors in sums.*
- [ex GiNaC::resultant](#) (const [ex](#) &e1, const [ex](#) &e2, const [ex](#) &s)  
*Resultant of two expressions  $e1, e2$  with respect to symbol  $s$ .*

### 9.67.1 Detailed Description

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 9.68 numeric.cpp File Reference

This file contains the interface to the underlying bignum package.

```
#include "numeric.h"
#include "ex.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <cln/output.h>
#include <cln/integer_io.h>
#include <cln/integer_ring.h>
#include <cln/rational_io.h>
#include <cln/rational_ring.h>
#include <cln/lfloat_class.h>
#include <cln/lfloat_io.h>
#include <cln/real_io.h>
#include <cln/real_ring.h>
#include <cln/complex_io.h>
#include <cln/complex_ring.h>
#include <cln/numtheory.h>
```

### Classes

- class [GiNaC::lanczos\\_coeffs](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([numeric](#), [basic](#), [print\\_func< print\\_context >\(&numeric::do\\_print\)](#), [print\\_func< print\\_latex >\(&numeric::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&numeric::do\\_print\\_csrc\)](#), [print\\_func< print\\_csrc\\_cl\\_N >\(&numeric::do\\_print\\_csrc\\_cl\\_N\)](#), [print\\_func< print\\_tree >\(&numeric::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&numeric::do\\_print\\_python\\_repr\)](#))  
[numeric](#)  
*default ctor.*
- static const [cln::cl\\_F](#) [GiNaC::make\\_real\\_float](#) (const [cln::cl\\_idecoded\\_float](#) &dec)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const [cln::cl\\_F](#) [GiNaC::read\\_real\\_float](#) (std::istream &s)  
*Read serialized floating point number.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([numeric](#))
- static void [GiNaC::write\\_real\\_float](#) (std::ostream &s, const [cln::cl\\_R](#) &n)
- static void [GiNaC::print\\_real\\_number](#) (const [print\\_context](#) &c, const [cln::cl\\_R](#) &x)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void [GiNaC::print\\_integer\\_csrc](#) (const [print\\_context](#) &c, const [cln::cl\\_I](#) &x)

*Helper function to print integer number in C++ source format.*

- static void `GiNaC::print_real_csrc` (const `print_context` &c, const `cln::cl_R` &x)

*Helper function to print real number in C++ source format.*

- template<typename T1, typename T2>  
static bool `GiNaC::coerce` (T1 &dst, const T2 &arg)
- template<> bool `GiNaC::coerce< int, cln::cl_I >` (int &dst, const `cln::cl_I` &arg)

*Check if CLN integer can be converted into int.*

- template<> bool `GiNaC::coerce< unsigned int, cln::cl_I >` (unsigned int &dst, const `cln::cl_I` &arg)
- static void `GiNaC::print_real_cl_N` (const `print_context` &c, const `cln::cl_R` &x)

*Helper function to print real number in C++ source format using `cl_N` types.*

- const `numeric` `GiNaC::exp` (const `numeric` &x)

*Exponential function.*

- const `numeric` `GiNaC::log` (const `numeric` &x)

*Natural logarithm.*

- const `numeric` `GiNaC::sin` (const `numeric` &x)

*Numeric sine (trigonometric function).*

- const `numeric` `GiNaC::cos` (const `numeric` &x)

*Numeric cosine (trigonometric function).*

- const `numeric` `GiNaC::tan` (const `numeric` &x)

*Numeric tangent (trigonometric function).*

- const `numeric` `GiNaC::asin` (const `numeric` &x)

*Numeric inverse sine (trigonometric function).*

- const `numeric` `GiNaC::acos` (const `numeric` &x)

*Numeric inverse cosine (trigonometric function).*

- const `numeric` `GiNaC::atan` (const `numeric` &x)

*Numeric arcustangent.*

- const `numeric` `GiNaC::atan` (const `numeric` &y, const `numeric` &x)

*Numeric arcustangent of two arguments, analytically continued in a suitable way.*

- const `numeric` `GiNaC::sinh` (const `numeric` &x)

*Numeric hyperbolic sine (trigonometric function).*

- const `numeric` `GiNaC::cosh` (const `numeric` &x)

*Numeric hyperbolic cosine (trigonometric function).*

- const `numeric` `GiNaC::tanh` (const `numeric` &x)

*Numeric hyperbolic tangent (trigonometric function).*

- const `numeric` `GiNaC::asinh` (const `numeric` &x)

*Numeric inverse hyperbolic sine (trigonometric function).*

- const `numeric` `GiNaC::acosh` (const `numeric` &x)

*Numeric inverse hyperbolic cosine (trigonometric function).*

- const `numeric` `GiNaC::atanh` (const `numeric` &x)

*Numeric inverse hyperbolic tangent (trigonometric function).*

- static `cln::cl_N` `GiNaC::Li2_series` (const `cln::cl_N` &x, const `cln::float_format_t` &prec)

*Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*

- static `cln::cl_N` `GiNaC::Li2_projection` (const `cln::cl_N` &x, const `cln::float_format_t` &prec)

*Folds `Li2`'s argument inside a small rectangle to enhance convergence.*

- const `cln::cl_N` `GiNaC::Li2_` (const `cln::cl_N` &value)

*Numeric evaluation of Dilogarithm.*

- const `numeric` `GiNaC::Li2` (const `numeric` &x)

- const `numeric` `GiNaC::zeta` (const `numeric` &x)

*Numeric evaluation of Riemann's Zeta function.*

- static `cln::float_format_t` `GiNaC::guess_precision` (const `cln::cl_N` &x)

- const `cln::cl_N` `GiNaC::lgamma` (const `cln::cl_N` &x)

*The Gamma function.*

- const `numeric GiNaC::lgamma` (const `numeric` &x)
- const `cln::cl_N GiNaC::tgamma` (const `cln::cl_N` &x)
- const `numeric GiNaC::tgamma` (const `numeric` &x)
- const `numeric GiNaC::psi` (const `numeric` &x)

*The psi function (aka polygamma function).*

- const `numeric GiNaC::psi` (const `numeric` &n, const `numeric` &x)

*The psi functions (aka polygamma functions).*

- const `numeric GiNaC::factorial` (const `numeric` &n)

*Factorial combinatorial function.*

- const `numeric GiNaC::doublefactorial` (const `numeric` &n)

*The double factorial combinatorial function.*

- const `numeric GiNaC::binomial` (const `numeric` &n, const `numeric` &k)

*The Binomial coefficients.*

- const `numeric GiNaC::bernoulli` (const `numeric` &nn)

*Bernoulli number.*

- const `numeric GiNaC::fibonacci` (const `numeric` &n)

*Fibonacci number.*

- const `numeric GiNaC::abs` (const `numeric` &x)

*Absolute value.*

- const `numeric GiNaC::mod` (const `numeric` &a, const `numeric` &b)

*Modulus (in positive representation).*

- const `numeric GiNaC::smod` (const `numeric` &a\_, const `numeric` &b\_)

*Modulus (in symmetric representation).*

- const `numeric GiNaC::irem` (const `numeric` &a, const `numeric` &b)

*Numeric integer remainder.*

- const `numeric GiNaC::irem` (const `numeric` &a, const `numeric` &b, `numeric` &q)

*Numeric integer remainder.*

- const `numeric GiNaC::iquo` (const `numeric` &a, const `numeric` &b)

*Numeric integer quotient.*

- const `numeric GiNaC::iquo` (const `numeric` &a, const `numeric` &b, `numeric` &r)

*Numeric integer quotient.*

- const `numeric GiNaC::gcd` (const `numeric` &a, const `numeric` &b)

*Greatest Common Divisor.*

- const `numeric GiNaC::lcm` (const `numeric` &a, const `numeric` &b)

*Least Common Multiple.*

- const `numeric GiNaC::sqrt` (const `numeric` &x)

*Numeric square root.*

- const `numeric GiNaC::isqrt` (const `numeric` &x)

*Integer numeric square root.*

- `ex GiNaC::PiEvalf` ()

*Floating point evaluation of Archimedes' constant Pi.*

- `ex GiNaC::EulerEvalf` ()

*Floating point evaluation of Euler's constant gamma.*

- `ex GiNaC::CatalanEvalf` ()

*Floating point evaluation of Catalan's constant.*

- `std::ostream & GiNaC::operator<<` (std::ostream &os, const `_numeric_digits` &e)

## Variables

- `const numeric GiNaC::I = numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))`  
*Imaginary unit.*
- `_numeric_digits GiNaC::Digits`  
*Accuracy in decimal digits.*

### 9.68.1 Detailed Description

This file contains the interface to the underlying bignum package.

Its most important design principle is to completely hide the inner working of that other package from the user of [GiNaC](#). It must either provide implementation of arithmetic operators and numerical evaluation of special functions or implement the interface to the bignum package.

## 9.69 numeric.h File Reference

Makes the interface to the underlying bignum package available.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <cln/complex.h>
#include <stdexcept>
#include <vector>
```

## Classes

- class [GiNaC::\\_numeric\\_digits](#)  
*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*
- class [GiNaC::pole\\_error](#)  
*Exception class thrown when a singularity is encountered.*
- class [GiNaC::numeric](#)  
*This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- `typedef void(* GiNaC::digits\_changed\_callback) (long)`  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`numeric`)
- `const numeric GiNaC::exp` (`const numeric &x`)  
*Exponential function.*
- `const numeric GiNaC::log` (`const numeric &x`)  
*Natural logarithm.*
- `const numeric GiNaC::sin` (`const numeric &x`)  
*Numeric sine (trigonometric function).*
- `const numeric GiNaC::cos` (`const numeric &x`)  
*Numeric cosine (trigonometric function).*
- `const numeric GiNaC::tan` (`const numeric &x`)  
*Numeric tangent (trigonometric function).*
- `const numeric GiNaC::asin` (`const numeric &x`)  
*Numeric inverse sine (trigonometric function).*
- `const numeric GiNaC::acos` (`const numeric &x`)  
*Numeric inverse cosine (trigonometric function).*
- `const numeric GiNaC::atan` (`const numeric &x`)  
*Numeric arcustangent.*
- `const numeric GiNaC::atan` (`const numeric &y`, `const numeric &x`)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- `const numeric GiNaC::sinh` (`const numeric &x`)  
*Numeric hyperbolic sine (trigonometric function).*
- `const numeric GiNaC::cosh` (`const numeric &x`)  
*Numeric hyperbolic cosine (trigonometric function).*
- `const numeric GiNaC::tanh` (`const numeric &x`)  
*Numeric hyperbolic tangent (trigonometric function).*
- `const numeric GiNaC::asinh` (`const numeric &x`)  
*Numeric inverse hyperbolic sine (trigonometric function).*
- `const numeric GiNaC::acosh` (`const numeric &x`)  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- `const numeric GiNaC::atanh` (`const numeric &x`)  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- `const numeric GiNaC::Li2` (`const numeric &x`)
- `const numeric GiNaC::zeta` (`const numeric &x`)  
*Numeric evaluation of Riemann's Zeta function.*
- `const numeric GiNaC::lgamma` (`const numeric &x`)
- `const numeric GiNaC::tgamma` (`const numeric &x`)
- `const numeric GiNaC::psi` (`const numeric &x`)  
*The psi function (aka polygamma function).*
- `const numeric GiNaC::psi` (`const numeric &n`, `const numeric &x`)  
*The psi functions (aka polygamma functions).*
- `const numeric GiNaC::factorial` (`const numeric &n`)  
*Factorial combinatorial function.*
- `const numeric GiNaC::doublefactorial` (`const numeric &n`)  
*The double factorial combinatorial function.*
- `const numeric GiNaC::binomial` (`const numeric &n`, `const numeric &k`)  
*The Binomial coefficients.*
- `const numeric GiNaC::bernoulli` (`const numeric &nn`)  
*Bernoulli number.*
- `const numeric GiNaC::fibonacci` (`const numeric &n`)

- Fibonacci number.*
  - `const numeric GiNaC::isqrt` (`const numeric &x`)
  - Integer numeric square root.*
  - `const numeric GiNaC::sqrt` (`const numeric &x`)
  - Numeric square root.*
  - `const numeric GiNaC::abs` (`const numeric &x`)
  - Absolute value.*
  - `const numeric GiNaC::mod` (`const numeric &a`, `const numeric &b`)
  - Modulus (in positive representation).*
  - `const numeric GiNaC::smod` (`const numeric &a_`, `const numeric &b_`)
  - Modulus (in symmetric representation).*
  - `const numeric GiNaC::irem` (`const numeric &a`, `const numeric &b`)
  - Numeric integer remainder.*
  - `const numeric GiNaC::irem` (`const numeric &a`, `const numeric &b`, `numeric &q`)
  - Numeric integer remainder.*
  - `const numeric GiNaC::iquo` (`const numeric &a`, `const numeric &b`)
  - Numeric integer quotient.*
  - `const numeric GiNaC::iquo` (`const numeric &a`, `const numeric &b`, `numeric &r`)
  - Numeric integer quotient.*
  - `const numeric GiNaC::gcd` (`const numeric &a`, `const numeric &b`)
  - Greatest Common Divisor.*
  - `const numeric GiNaC::lcm` (`const numeric &a`, `const numeric &b`)
  - Least Common Multiple.*
  - `const numeric GiNaC::pow` (`const numeric &x`, `const numeric &y`)
  - `const numeric GiNaC::inverse` (`const numeric &x`)
  - `numeric GiNaC::step` (`const numeric &x`)
  - `int GiNaC::csgn` (`const numeric &x`)
  - `bool GiNaC::is_zero` (`const numeric &x`)
  - `bool GiNaC::is_positive` (`const numeric &x`)
  - `bool GiNaC::is_negative` (`const numeric &x`)
  - `bool GiNaC::is_integer` (`const numeric &x`)
  - `bool GiNaC::is_pos_integer` (`const numeric &x`)
  - `bool GiNaC::is_nonneg_integer` (`const numeric &x`)
  - `bool GiNaC::is_even` (`const numeric &x`)
  - `bool GiNaC::is_odd` (`const numeric &x`)
  - `bool GiNaC::is_prime` (`const numeric &x`)
  - `bool GiNaC::is_rational` (`const numeric &x`)
  - `bool GiNaC::is_real` (`const numeric &x`)
  - `bool GiNaC::is_cinteger` (`const numeric &x`)
  - `bool GiNaC::is_crational` (`const numeric &x`)
  - `int GiNaC::to_int` (`const numeric &x`)
  - `long GiNaC::to_long` (`const numeric &x`)
  - `double GiNaC::to_double` (`const numeric &x`)
  - `const numeric GiNaC::real` (`const numeric &x`)
  - `const numeric GiNaC::imag` (`const numeric &x`)
  - `const numeric GiNaC::numer` (`const numeric &x`)
  - `const numeric GiNaC::denom` (`const numeric &x`)
  - `ex GiNaC::PiEvalf` ()
  - Floating point evaluation of Archimedes' constant Pi.*
  - `ex GiNaC::EulerEvalf` ()
  - Floating point evaluation of Euler's constant gamma.*
  - `ex GiNaC::CatalanEvalf` ()
  - Floating point evaluation of Catalan's constant.*



### 9.69.1 Detailed Description

Makes the interface to the underlying bignum package available.

## 9.70 operators.cpp File Reference

Implementation of [GiNaC](#)'s overloaded operators.

```
#include "operators.h"
#include "numeric.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "ncmul.h"
#include "relational.h"
#include "print.h"
#include "utils.h"
```

### Namespaces

- namespace [GiNaC](#)

### Enumerations

- enum { [GiNaC::callback\\_registered](#) = 1 }

### Functions

- static const [ex](#) [GiNaC::exadd](#) (const [ex](#) &lh, const [ex](#) &rh)  
*Used internally by [operator+](#)() to add two ex objects.*
- static const [ex](#) [GiNaC::exmul](#) (const [ex](#) &lh, const [ex](#) &rh)  
*Used internally by [operator\\*](#)() to multiply two ex objects.*
- static const [ex](#) [GiNaC::exminus](#) (const [ex](#) &lh)  
*Used internally by [operator-](#)() and friends to change the sign of an argument.*
- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator\\*](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator/](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator\\*](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator/](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- [ex](#) & [GiNaC::operator+=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator-=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator\\*=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator/=](#) ([ex](#) &lh, const [ex](#) &rh)
- [numeric](#) & [GiNaC::operator+=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator-=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator\\*=](#) ([numeric](#) &lh, const [numeric](#) &rh)

- `numeric & GiNaC::operator/= (numeric &lh, const numeric &rh)`
- `const ex GiNaC::operator+ (const ex &lh)`
- `const ex GiNaC::operator- (const ex &lh)`
- `const numeric GiNaC::operator+ (const numeric &lh)`
- `const numeric GiNaC::operator- (const numeric &lh)`
- `ex & GiNaC::operator++ (ex &rh)`  
*Expression prefix increment.*
- `ex & GiNaC::operator-- (ex &rh)`  
*Expression prefix decrement.*
- `const ex GiNaC::operator++ (ex &lh, int)`  
*Expression postfix increment.*
- `const ex GiNaC::operator-- (ex &lh, int)`  
*Expression postfix decrement.*
- `numeric & GiNaC::operator++ (numeric &rh)`  
*Numeric prefix increment.*
- `numeric & GiNaC::operator-- (numeric &rh)`  
*Numeric prefix decrement.*
- `const numeric GiNaC::operator++ (numeric &lh, int)`  
*Numeric postfix increment.*
- `const numeric GiNaC::operator-- (numeric &lh, int)`  
*Numeric postfix decrement.*
- `const relational GiNaC::operator== (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator!= (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator< (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator<= (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator> (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator>= (const ex &lh, const ex &rh)`
- `static int GiNaC::my_ios_index ()`
- `static void GiNaC::my_ios_callback (std::ios_base::event ev, std::ios_base &s, int i)`
- `static print_context * GiNaC::get_print_context (std::ios_base &s)`
- `static void GiNaC::set_print_context (std::ios_base &s, const print_context &c)`
- `static unsigned GiNaC::get_print_options (std::ios_base &s)`
- `static void GiNaC::set_print_options (std::ostream &s, unsigned options)`
- `std::ostream & GiNaC::operator<< (std::ostream &os, const ex &e)`
- `std::ostream & GiNaC::operator<< (std::ostream &os, const exvector &e)`
- `std::ostream & GiNaC::operator<< (std::ostream &os, const exset &e)`
- `std::ostream & GiNaC::operator<< (std::ostream &os, const exmap &e)`
- `std::istream & GiNaC::operator>> (std::istream &is, ex &e)`
- `std::ostream & GiNaC::dflt (std::ostream &os)`
- `std::ostream & GiNaC::latex (std::ostream &os)`
- `std::ostream & GiNaC::python (std::ostream &os)`
- `std::ostream & GiNaC::python_repr (std::ostream &os)`
- `std::ostream & GiNaC::tree (std::ostream &os)`
- `std::ostream & GiNaC::csrc (std::ostream &os)`
- `std::ostream & GiNaC::csrc_float (std::ostream &os)`
- `std::ostream & GiNaC::csrc_double (std::ostream &os)`
- `std::ostream & GiNaC::csrc_cl_N (std::ostream &os)`
- `std::ostream & GiNaC::index_dimensions (std::ostream &os)`
- `std::ostream & GiNaC::no_index_dimensions (std::ostream &os)`

### 9.70.1 Detailed Description

Implementation of `GiNaC`'s overloaded operators.

## 9.71 operators.h File Reference

Interface to [GiNaC](#)'s overloaded operators.

```
#include <iosfwd>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator\\*](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator/](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator\\*](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator/](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- [ex](#) & [GiNaC::operator+=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator-=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator\\*=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator/=](#) ([ex](#) &lh, const [ex](#) &rh)
- [numeric](#) & [GiNaC::operator+=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator-=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator\\*=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator/=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh)
- [ex](#) & [GiNaC::operator++](#) ([ex](#) &rh)  
*Expression prefix increment.*
- [ex](#) & [GiNaC::operator--](#) ([ex](#) &rh)  
*Expression prefix decrement.*
- const [ex](#) [GiNaC::operator++](#) ([ex](#) &lh, int)  
*Expression postfix increment.*
- const [ex](#) [GiNaC::operator--](#) ([ex](#) &lh, int)  
*Expression postfix decrement.*
- [numeric](#) & [GiNaC::operator++](#) ([numeric](#) &rh)  
*Numeric prefix increment.*
- [numeric](#) & [GiNaC::operator--](#) ([numeric](#) &rh)  
*Numeric prefix decrement.*
- const [numeric](#) [GiNaC::operator++](#) ([numeric](#) &lh, int)  
*Numeric postfix increment.*
- const [numeric](#) [GiNaC::operator--](#) ([numeric](#) &lh, int)  
*Numeric postfix decrement.*
- const [relational](#) [GiNaC::operator==](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator!=](#) (const [ex](#) &lh, const [ex](#) &rh)

- const relational `GiNaC::operator<` (const `ex` &lh, const `ex` &rh)
- const relational `GiNaC::operator<=` (const `ex` &lh, const `ex` &rh)
- const relational `GiNaC::operator>` (const `ex` &lh, const `ex` &rh)
- const relational `GiNaC::operator>=` (const `ex` &lh, const `ex` &rh)
- `std::ostream` & `GiNaC::operator<<` (`std::ostream` &os, const `ex` &e)
- `std::istream` & `GiNaC::operator>>` (`std::istream` &is, `ex` &e)
- `std::ostream` & `GiNaC::dflt` (`std::ostream` &os)
- `std::ostream` & `GiNaC::latex` (`std::ostream` &os)
- `std::ostream` & `GiNaC::python` (`std::ostream` &os)
- `std::ostream` & `GiNaC::python_repr` (`std::ostream` &os)
- `std::ostream` & `GiNaC::tree` (`std::ostream` &os)
- `std::ostream` & `GiNaC::csrc` (`std::ostream` &os)
- `std::ostream` & `GiNaC::csrc_float` (`std::ostream` &os)
- `std::ostream` & `GiNaC::csrc_double` (`std::ostream` &os)
- `std::ostream` & `GiNaC::csrc_cl_N` (`std::ostream` &os)
- `std::ostream` & `GiNaC::index_dimensions` (`std::ostream` &os)
- `std::ostream` & `GiNaC::no_index_dimensions` (`std::ostream` &os)

### 9.71.1 Detailed Description

Interface to `GiNaC`'s overloaded operators.

## 9.72 `power.cpp` File Reference

Implementation of `GiNaC`'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "power.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "constant.h"
#include "operators.h"
#include "inifcns.h"
#include "matrix.h"
#include "indexed.h"
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "relational.h"
#include "compiler.h"
#include <limits>
#include <stdexcept>
#include <vector>
#include <algorithm>
```

### Namespaces

- namespace `GiNaC`

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`power`, `basic`, `print_func< print_dflt >(&power::do_print_dflt)`, `print_func< print_latex >(&power::do_print_latex)`, `print_func< print_csrc >(&power::do_print_csrc)`, `print_func< print_python >(&power::do_print_python)`, `print_func< print_python_repr >(&power::do_print_python_repr)`, `print_func< print_csrc_cl_N >(&power::do_print_csrc_cl_N)`) `power`
- static void `GiNaC::print_sym_pow` (const `print_context` &c, const `symbol` &x, int `exp`)
- bool `GiNaC::tryfactsubs` (const `ex` &origfactor, const `ex` &patternfactor, int &nummatches, `exmap` &repls)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`power`)

### 9.72.1 Detailed Description

Implementation of `GiNaC`'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 9.73 power.h File Reference

Interface to `GiNaC`'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

## Classes

- class `GiNaC::power`  
*This class holds a two-component object, a basis and an exponent representing exponentiation.*

## Namespaces

- namespace `GiNaC`

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`power`)
- `ex GiNaC::pow` (const `ex` &b, const `ex` &e)  
*Symbolic exponentiation.*
- `template<typename T1, typename T2> ex GiNaC::pow` (const T1 &b, const T2 &e)
- `ex GiNaC::sqrt` (const `ex` &a)  
*Square root expression.*

### 9.73.1 Detailed Description

Interface to `GiNaC`'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 9.74 print.cpp File Reference

Implementation of helper classes for expression output.

```
#include "print.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

### Variables

- unsigned [GiNaC::next\\_print\\_context\\_id](#) = 0  
*Next free ID for [print\\_context](#) types.*

### 9.74.1 Detailed Description

Implementation of helper classes for expression output.

## 9.75 print.h File Reference

Definition of helper classes for expression output.

```
#include "class_info.h"
#include <iosfwd>
#include <memory>
```

### Classes

- class [GiNaC::print\\_context\\_options](#)  
*This class stores information about a registered [print\\_context](#) class.*
- class [GiNaC::print\\_options](#)  
*Flags to control the behavior of a [print\\_context](#).*
- class [GiNaC::print\\_context](#)  
*Base class for [print\\_contexts](#).*
- class [GiNaC::print\\_dflt](#)  
*Context for default (ginsh-parsable) output.*
- class [GiNaC::print\\_latex](#)  
*Context for latex-parsable output.*
- class [GiNaC::print\\_python](#)  
*Context for python pretty-print output.*
- class [GiNaC::print\\_python\\_repr](#)  
*Context for python-parsable output.*
- class [GiNaC::print\\_tree](#)

- Context for tree-like output for debugging.*

  - class [GiNaC::print\\_csrc](#)

*Base context for C source output.*
  - class [GiNaC::print\\_csrc\\_float](#)

*Context for C source output using float precision.*
  - class [GiNaC::print\\_csrc\\_double](#)

*Context for C source output using double precision.*
  - class [GiNaC::print\\_csrc\\_cl\\_N](#)

*Context for C source output using CLN numbers.*
  - class [GiNaC::print\\_functor\\_impl](#)

*Base class for [print\\_functor](#) handlers.*
  - class [GiNaC::print\\_ptrfun\\_handler< T, C >](#)

*[print\\_functor](#) handler for pointer-to-functions of class T, context type C*
  - class [GiNaC::print\\_memfun\\_handler< T, C >](#)

*[print\\_functor](#) handler for member functions of class T, context type C*
  - class [GiNaC::print\\_functor](#)

*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define [GINAC\\_DECLARE\\_PRINT\\_CONTEXT\\_COMMON](#)(classname)

*Common part of [GINAC\\_DECLARE\\_PRINT\\_CONTEXT\\_BASE](#) and [GINAC\\_DECLARE\\_PRINT\\_CONTEXT\\_DERIVED](#).*
- #define [GINAC\\_DECLARE\\_PRINT\\_CONTEXT\\_BASE](#)(classname)
- #define [GINAC\\_DECLARE\\_PRINT\\_CONTEXT](#)(classname, supertype)

*Macro for inclusion in the declaration of a [print\\_context](#) class.*
- #define [GINAC\\_IMPLEMENT\\_PRINT\\_CONTEXT](#)(classname, supertype)

*Macro for inclusion in the implementation of each [print\\_context](#) class.*

## Typedefs

- typedef [class\\_info< print\\_context\\_options > GiNaC::print\\_context\\_class\\_info](#)

## Functions

- template<class T>  
bool [GiNaC::is\\_a](#) (const [print\\_context](#) &obj)

*Check if obj is a T, including base classes.*

### 9.75.1 Detailed Description

Definition of helper classes for expression output.

## 9.75.2 Macro Definition Documentation

### 9.75.2.1 GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON

```
#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(  
    classname)
```

**Value:**

```
public: \  
    friend class function_options; \  
    friend class registered_class_options; \  
    static const GiNaC::print_context_class_info &get_class_info_static(); \  
    classname();
```

Common part of GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE and GINAC\_DECLARE\_PRINT\_CONTEXT\_DERIVED.

### 9.75.2.2 GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE

```
#define GINAC_DECLARE_PRINT_CONTEXT_BASE(  
    classname)
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \  
virtual const GiNaC::print_context_class_info &get_class_info() const { return  
    classname::get_class_info_static(); } \  
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); }  
\  
virtual classname * duplicate() const { return new classname(*this); } \  
private:
```

### 9.75.2.3 GINAC\_DECLARE\_PRINT\_CONTEXT

```
#define GINAC_DECLARE_PRINT_CONTEXT(  
    classname,  
    supertype)
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \  
typedef supertype inherited; \  
const GiNaC::print_context_class_info &get_class_info() const override { return  
    classname::get_class_info_static(); } \  
const char *class_name() const override { return classname::get_class_info_static().options.get_name();  
} \  
classname * duplicate() const override { return new classname(*this); } \  
private:
```

Macro for inclusion in the declaration of a print\_context class.

It declares some functions that are common to all classes derived from 'print\_context' as well as all required stuff for the [GiNaC](#) registry.



### 9.75.2.4 GINAC\_IMPLEMENT\_PRINT\_CONTEXT

```
#define GINAC_IMPLEMENT_PRINT_CONTEXT(
    classname,
    surname)
```

#### Value:

```
const GiNaC::print_context_class_info &classname::get_class_info_static() \
{ \
    static GiNaC::print_context_class_info reg_info = \
        GiNaC::print_context_class_info(GiNaC::print_context_options(#classname, #surname, \
        GiNaC::next_print_context_id++)); \
    return reg_info; \
}
```

Macro for inclusion in the implementation of each `print_context` class.

## 9.76 pseries.cpp File Reference

Implementation of class for extended truncated power series and methods for series expansion.

```
#include "pseries.h"
#include "add.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "integral.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <numeric>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([pseries](#), [basic](#), [print\\_func< print\\_context >\(&pseries::do\\_print\)](#), [print\\_func< print\\_latex >\(&pseries::do\\_print\\_latex\)](#), [print\\_func< print\\_tree >\(&pseries::do\\_print\\_tree\)](#), [print\\_func< print\\_python >\(&pseries::do\\_print\\_python\)](#), [print\\_func< print\\_python\\_repr >\(&pseries::do\\_print\\_python\\_repr\)](#)) [pseries](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))

### 9.76.1 Detailed Description

Implementation of class for extended truncated power series and methods for series expansion.

## 9.77 pseries.h File Reference

Interface to class for extended truncated power series.

```
#include "basic.h"
#include "expairseq.h"
```

### Classes

- class [GiNaC::pseries](#)  
*This class holds a extended truncated power series (positive and negative integer powers).*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex GiNaC::series\\_to\\_poly](#) (const [ex](#) &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [GiNaC::is\\_terminating](#) (const [pseries](#) &s)

### 9.77.1 Detailed Description

Interface to class for extended truncated power series.

## 9.78 ptr.h File Reference

Reference-counted pointer template.

```
#include "assertion.h"
#include <cstddef>
#include <functional>
#include <iosfwd>
```

### Classes

- class [GiNaC::refcounted](#)  
*Base class for reference-counted objects.*
- class [GiNaC::ptr< T >](#)  
*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
- struct [std::less< GiNaC::ptr< T > >](#)  
*Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.*

## Namespaces

- namespace [GiNaC](#)
- namespace [std](#)

### 9.78.1 Detailed Description

Reference-counted pointer template.

## 9.79 registrar.cpp File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "registrar.h"
```

## Namespaces

- namespace [GiNaC](#)

### 9.79.1 Detailed Description

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

## 9.80 registrar.h File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "class_info.h"
#include "print.h"
#include <list>
#include <typeinfo>
#include <vector>
```

## Classes

- struct [GiNaC::return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [GiNaC::registered\\_class\\_options](#)  
*This class stores information about a registered [GiNaC](#) class.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname)`  
Common part of `GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS` and `GINAC_DECLARE_REGISTERED_CLASS`.
- `#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(classname, supname)`  
Primary macro for inclusion in the declaration of each registered class.
- `#define GINAC_DECLARE_REGISTERED_CLASS(classname, supname)`  
Macro for inclusion in the declaration of each registered class.
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS(classname, supname)`  
Macro for inclusion in the implementation of each registered class.
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(classname, supname, options)`  
Macro for inclusion in the implementation of each registered class.
- `#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(classname, supname, options)`  
Macro for inclusion in the implementation of each registered class.

## Typedefs

- `typedef class_info< registered_class_options > GiNaC::registered_class_info`

## Functions

- `template<typename T>`  
`return_type_t GiNaC::make_return_type_t (const unsigned rl=0)`
- `template<class Alg, class Ctx, class T, class C>`  
`void GiNaC::set_print_func (void f(const T &, const C &c, unsigned))`  
Add or replace a print method.
- `template<class Alg, class Ctx, class T, class C>`  
`void GiNaC::set_print_func (void(T::*f)(const C &, unsigned))`  
Add or replace a print method.

### 9.80.1 Detailed Description

`GiNaC`'s class registrar (for class basic and all classes derived from it).

### 9.80.2 Macro Definition Documentation

#### 9.80.2.1 GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON

```
#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(  
    classname)
```

#### Value:

```
private: \
    static GiNaC::registered_class_info reg_info; \
public: \
    static GiNaC::registered_class_info &get_class_info_static() { return reg_info; } \
    class visitor { \
    public: \
        virtual void visit(const classname &) = 0; \
        virtual ~visitor() {}; \
    };
```

Common part of `GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS` and `GINAC_DECLARE_REGISTERED_CLASS`.

### 9.80.2.2 GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS

```
#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(  
    classname,  
    supertype)
```

#### Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \  
typedef supertype inherited; \  
virtual const GiNaC::registered_class_info &get_class_info() const { return  
    classname::get_class_info_static(); } \  
virtual GiNaC::registered_class_info &get_class_info() { return classname::get_class_info_static(); } \  
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); } \  
private:
```

Primary macro for inclusion in the declaration of each registered class.

### 9.80.2.3 GINAC\_DECLARE\_REGISTERED\_CLASS

```
#define GINAC_DECLARE_REGISTERED_CLASS(  
    classname,  
    supertype)
```

#### Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \  
template<class B, typename... Args> friend B & dynallocate(Args &&... args); \  
typedef supertype inherited; \  
classname(); \  
classname * duplicate() const override { \  
    classname * bp = new classname(*this); \  
    bp->setflag(GiNaC::status_flags::dynallocated); \  
    return bp; \  
} \  
void accept(GiNaC::visitor & v) const override \  
{ \  
    if (visitor *p = dynamic_cast<visitor *>(&v)) \  
        p->visit(*this); \  
    else \  
        inherited::accept(v); \  
} \  
const GiNaC::registered_class_info &get_class_info() const override { return  
    classname::get_class_info_static(); } \  
GiNaC::registered_class_info &get_class_info() override { return classname::get_class_info_static(); } \  
const char *class_name() const override { return classname::get_class_info_static().options.get_name(); } \  
protected: \  
    int compare_same_type(const GiNaC::basic & other) const override; \  
private:
```

Macro for inclusion in the declaration of each registered class.

It declares some functions that are common to all classes derived from 'basic' as well as all required stuff for the [GiNaC](#) class registry (mainly needed for archiving).

### 9.80.2.4 GINAC\_IMPLEMENT\_REGISTERED\_CLASS

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS(  
    classname,  
    supertype)
```

#### Value:

```
GiNaC::registered_class_info classname::reg_info =  
    GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supertype,  
        typeid(classname)));
```

Macro for inclusion in the implementation of each registered class.

### 9.80.2.5 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(  
    classname,  
    supername,  
    options)
```

#### Value:

```
GiNaC::registered_class_info classname::reg_info =  
    GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supername,  
        typeid(classname).options);
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

### 9.80.2.6 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(  
    classname,  
    supername,  
    options)
```

#### Value:

```
GiNaC::registered_class_info classname::reg_info =  
    GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supername,  
        typeid(classname).options);
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

## 9.81 relational.cpp File Reference

Implementation of relations between expressions.

```
#include "relational.h"  
#include "operators.h"  
#include "numeric.h"  
#include "archive.h"  
#include "utils.h"  
#include "hash_seed.h"  
#include <stdexcept>
```

#### Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([relational](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[relational::do\\_print](#)), [print\\_func](#)< [print\\_tree](#)>(&[relational::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#)>(&[relational::do\\_print\\_python\\_repr](#))) [relational](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([relational](#))
- static void [GiNaC::print\\_operator](#) (const [print\\_context](#) &c, [relational::operators](#) o)

### 9.81.1 Detailed Description

Implementation of relations between expressions.

## 9.82 relational.h File Reference

Interface to relations between expressions.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::relational](#)  
*This class holds a relation consisting of two expressions and a logical relation between them.*
- struct [GiNaC::relational::safe\\_bool\\_helper](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([relational](#))

### 9.82.1 Detailed Description

Interface to relations between expressions.

## 9.83 remember.cpp File Reference

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

```
#include "function.h"
#include "utils.h"
#include "remember.h"
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

### 9.83.1 Detailed Description

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

## 9.84 remember.h File Reference

Interface to helper classes for using the remember option in [GiNaC](#) functions.

```
#include <iosfwd>
#include <list>
#include <vector>
```

## Classes

- class [GiNaC::remember\\_table\\_entry](#)  
*A single entry in the remember table of a function.*
- class [GiNaC::remember\\_table\\_list](#)  
*A list of entries in the remember table having some least significant bits of the hashvalue in common.*
- class [GiNaC::remember\\_table](#)  
*The remember table is organized like an n-fold associative cache in a microprocessor.*

## Namespaces

- namespace [GiNaC](#)

### 9.84.1 Detailed Description

Interface to helper classes for using the remember option in [GiNaC](#) functions.

## 9.85 structure.h File Reference

Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include "ex.h"
#include "ncmul.h"
#include "numeric.h"
#include "operators.h"
#include "print.h"
#include <functional>
```



## Classes

- class [GiNaC::compare\\_all\\_equal< T >](#)  
*Comparison policy: all structures of one type are equal.*
- class [GiNaC::compare\\_std\\_less< T >](#)  
*Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.*
- class [GiNaC::compare\\_bitwise< T >](#)  
*Comparison policy: use bit-wise comparison to compare structures.*
- class [GiNaC::structure< T, ComparisonPolicy >](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*

## Namespaces

- namespace [GiNaC](#)

## Variables

- `template<class T, template< class > class CP>`  
`registered_class_info GiNaC::structure< T, CP >::reg_info = registered_class_info(registered_class_options(structure::get_cla`  
`"basic", typeid(structure<T, CP>)))`

### 9.85.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of C++ structures.

## 9.86 symbol.cpp File Reference

Implementation of [GiNaC](#)'s symbolic objects.

```
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <map>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symbol](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symbol::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[symbol::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[symbol::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[symbol::do\\_print\\_python\\_repr](#))) [symbol](#)
- static const std::string & [GiNaC::get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([symbol](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([realsymbol](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([possymbol](#))

### 9.86.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic objects.

## 9.87 symbol.h File Reference

Interface to [GiNaC](#)'s symbolic objects.

```
#include "basic.h"
#include "ex.h"
#include "ptr.h"
#include "archive.h"
#include <string>
#include <typeinfo>
```

## Classes

- class [GiNaC::symbol](#)  
*Basic CAS symbol.*
- class [GiNaC::realsymbol](#)  
*Specialization of symbol to real domain.*
- class [GiNaC::possymbol](#)  
*Specialization of symbol to real positive domain.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([symbol](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([realsymbol](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([possymbol](#))

### 9.87.1 Detailed Description

Interface to [GiNaC](#)'s symbolic objects.

## 9.88 symmetry.cpp File Reference

Implementation of [GiNaC](#)'s symmetry definitions.

```
#include "symmetry.h"
#include "lst.h"
#include "add.h"
#include "numeric.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <functional>
#include <limits>
#include <stdexcept>
```

### Classes

- class [GiNaC::sy\\_is\\_less](#)
- class [GiNaC::sy\\_swap](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symmetry](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symmetry::do\\_print](#)), [print\\_func](#)< [print\\_tree](#) >(&[symmetry::do\\_print\\_tree](#))) [symmetry](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([symmetry](#))
- static const [symmetry](#) & [GiNaC::index0](#) ()
- static const [symmetry](#) & [GiNaC::index1](#) ()
- static const [symmetry](#) & [GiNaC::index2](#) ()
- static const [symmetry](#) & [GiNaC::index3](#) ()
- const [symmetry](#) & [GiNaC::not\\_symmetric](#) ()
- const [symmetry](#) & [GiNaC::symmetric2](#) ()
- const [symmetry](#) & [GiNaC::symmetric3](#) ()
- const [symmetry](#) & [GiNaC::symmetric4](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric2](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric3](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) ([exvector::iterator](#) v, const [symmetry](#) &[symm](#))  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static [ex](#) [GiNaC::symm](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) [first](#), [exvector::const\\_iterator](#) last, bool asymmetric)
- [ex](#) [GiNaC::symmetrize](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) [first](#), [exvector::const\\_iterator](#) last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::antisymmetrize](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) [first](#), [exvector::const\\_iterator](#) last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::symmetrize\\_cyclic](#) (const [ex](#) &[e](#), [exvector::const\\_iterator](#) [first](#), [exvector::const\\_iterator](#) last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 9.88.1 Detailed Description

Implementation of [GiNaC](#)'s symmetry definitions.

## 9.89 symmetry.h File Reference

Interface to [GiNaC](#)'s symmetry definitions.

```
#include "ex.h"
#include "archive.h"
#include <set>
```

### Classes

- class [GiNaC::symmetry](#)  
*This class describes the symmetry of a group of indices.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([symmetry](#))
- [symmetry](#) [GiNaC::sy\\_none](#) ()
- [symmetry](#) [GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry](#) [GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry](#) [GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry](#) [GiNaC::sy\\_symm](#) ()
- [symmetry](#) [GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry](#) [GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry](#) [GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry](#) [GiNaC::sy\\_anti](#) ()
- [symmetry](#) [GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry](#) [GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry](#) [GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry](#) [GiNaC::sy\\_cycl](#) ()
- [symmetry](#) [GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry](#) [GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry](#) [GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- const [symmetry](#) & [GiNaC::not\\_symmetric](#) ()
- const [symmetry](#) & [GiNaC::symmetric2](#) ()
- const [symmetry](#) & [GiNaC::symmetric3](#) ()
- const [symmetry](#) & [GiNaC::symmetric4](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric2](#) ()

- const [symmetry](#) & [GiNaC::antisymmetric3](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)
 

*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- [ex](#) [GiNaC::symmetrize](#) (const [ex](#) &e, exvector::const\_iterator [first](#), exvector::const\_iterator last)
 

*Symmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)
 

*Symmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::antisymmetrize](#) (const [ex](#) &e, exvector::const\_iterator [first](#), exvector::const\_iterator last)
 

*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::antisymmetrize](#) (const [ex](#) &e, const [exvector](#) &v)
 

*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::symmetrize\\_cyclic](#) (const [ex](#) &e, exvector::const\_iterator [first](#), exvector::const\_iterator last)
 

*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [ex](#) [GiNaC::symmetrize\\_cyclic](#) (const [ex](#) &e, const [exvector](#) &v)
 

*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 9.89.1 Detailed Description

Interface to [GiNaC](#)'s symmetry definitions.

## 9.90 tensor.cpp File Reference

Implementation of [GiNaC](#)'s special tensors.

```
#include "tensor.h"
#include "idx.h"
#include "indexed.h"
#include "symmetry.h"
#include "relational.h"
#include "operators.h"
#include "lst.h"
#include "numeric.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (tensdelta, tensor, print_func< print_dflt >(&tensdelta::do_print). print_func< print_latex >(&tensdelta::do_print_latex))` `GINAC_IMPLEMENT_↵`  
`REGISTERED_CLASS_OPT(tensmetric`
- `GiNaC::print_func< print_dflt > (&tensmetric::do_print). print_func< print_latex >(&tensmetric`
- `GiNaC::GINAC_BIND_UNARCHIVER (minkmetric)`
- `GiNaC::GINAC_BIND_UNARCHIVER (tensepsilon)`
- `GiNaC::GINAC_BIND_UNARCHIVER (tensdelta)`
- `GiNaC::GINAC_BIND_UNARCHIVER (tensmetric)`
- `GiNaC::GINAC_BIND_UNARCHIVER (spinmetric)`
- `ex GiNaC::delta_tensor (const ex &i1, const ex &i2)`  
*Create a delta tensor with specified indices.*
- `ex GiNaC::metric_tensor (const ex &i1, const ex &i2)`  
*Create a symmetric metric tensor with specified indices.*
- `ex GiNaC::lorentz_g (const ex &i1, const ex &i2, bool pos_sig=false)`  
*Create a Minkowski metric tensor with specified indices.*
- `ex GiNaC::spinor_metric (const ex &i1, const ex &i2)`  
*Create a spinor metric tensor with specified indices.*
- `ex GiNaC::epsilon_tensor (const ex &i1, const ex &i2)`  
*Create an epsilon tensor in a Euclidean space with two indices.*
- `ex GiNaC::epsilon_tensor (const ex &i1, const ex &i2, const ex &i3)`  
*Create an epsilon tensor in a Euclidean space with three indices.*
- `ex GiNaC::lorentz_eps (const ex &i1, const ex &i2, const ex &i3, const ex &i4, bool pos_sig=false)`  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 9.90.1 Detailed Description

Implementation of `GiNaC`'s special tensors.

## 9.91 tensor.h File Reference

Interface to `GiNaC`'s special tensors.

```
#include "ex.h"
#include "archive.h"
```

## Classes

- class `GiNaC::tensor`  
*This class holds one of `GiNaC`'s predefined special tensors such as the delta and the metric tensors.*
- class `GiNaC::tensdelta`  
*This class represents the delta tensor.*
- class `GiNaC::tensmetric`  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class `GiNaC::minkmetric`  
*This class represents a Minkowski metric tensor.*
- class `GiNaC::spinmetric`  
*This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*
- class `GiNaC::tensepsilon`  
*This class represents the totally antisymmetric epsilon tensor.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensdelta](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([minkmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensepsilon](#))
- [ex GiNaC::delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex GiNaC::metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex GiNaC::lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex GiNaC::spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex GiNaC::lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 9.91.1 Detailed Description

Interface to [GiNaC](#)'s special tensors.

## 9.92 utils.cpp File Reference

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "ex.h"
#include "numeric.h"
#include "utils.h"
#include "version.h"
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- unsigned [GiNaC::log2](#) (unsigned n)  
*Integer binary logarithm.*
- const [numeric GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient n!*

## Variables

- const int [GiNaC::version\\_major](#) = GINACLIB\_MAJOR\_VERSION
- const int [GiNaC::version\\_minor](#) = GINACLIB\_MINOR\_VERSION
- const int [GiNaC::version\\_micro](#) = GINACLIB\_MICRO\_VERSION
- const [numeric](#) \* [GiNaC::\\_num\\_120\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_120](#) = [ex](#)(\*[\\_num\\_120\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_60\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_60](#) = [ex](#)(\*[\\_num\\_60\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_48\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_48](#) = [ex](#)(\*[\\_num\\_48\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_30\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_30](#) = [ex](#)(\*[\\_num\\_30\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_25\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_25](#) = [ex](#)(\*[\\_num\\_25\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_24\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_24](#) = [ex](#)(\*[\\_num\\_24\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_20\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_20](#) = [ex](#)(\*[\\_num\\_20\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_18\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_18](#) = [ex](#)(\*[\\_num\\_18\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_15\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_15](#) = [ex](#)(\*[\\_num\\_15\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_12\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_12](#) = [ex](#)(\*[\\_num\\_12\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_11\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_11](#) = [ex](#)(\*[\\_num\\_11\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_10\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_10](#) = [ex](#)(\*[\\_num\\_10\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_9\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_9](#) = [ex](#)(\*[\\_num\\_9\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_8\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_8](#) = [ex](#)(\*[\\_num\\_8\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_7\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_7](#) = [ex](#)(\*[\\_num\\_7\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_6\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_6](#) = [ex](#)(\*[\\_num\\_6\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_5\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_5](#) = [ex](#)(\*[\\_num\\_5\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_4\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_4](#) = [ex](#)(\*[\\_num\\_4\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_3\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_3](#) = [ex](#)(\*[\\_num\\_3\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_2\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_2](#) = [ex](#)(\*[\\_num\\_2\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_1\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_1](#) = [ex](#)(\*[\\_num\\_1\\_p](#))



- `const numeric * GiNaC::_num_1_2_p`
- `const ex GiNaC::_ex_1_2 = ex(*_num_1_2_p)`
- `const numeric * GiNaC::_num_1_3_p`
- `const ex GiNaC::_ex_1_3 = ex(*_num_1_3_p)`
- `const numeric * GiNaC::_num_1_4_p`
- `const ex GiNaC::_ex_1_4 = ex(*_num_1_4_p)`
- `const numeric * GiNaC::_num0_p`
- `const ex GiNaC::_ex0 = ex(*_num0_p)`
- `const numeric * GiNaC::_num1_4_p`
- `const ex GiNaC::_ex1_4 = ex(*_num1_4_p)`
- `const numeric * GiNaC::_num1_3_p`
- `const ex GiNaC::_ex1_3 = ex(*_num1_3_p)`
- `const numeric * GiNaC::_num1_2_p`
- `const ex GiNaC::_ex1_2 = ex(*_num1_2_p)`
- `const numeric * GiNaC::_num1_p`
- `const ex GiNaC::_ex1 = ex(*_num1_p)`
- `const numeric * GiNaC::_num2_p`
- `const ex GiNaC::_ex2 = ex(*_num2_p)`
- `const numeric * GiNaC::_num3_p`
- `const ex GiNaC::_ex3 = ex(*_num3_p)`
- `const numeric * GiNaC::_num4_p`
- `const ex GiNaC::_ex4 = ex(*_num4_p)`
- `const numeric * GiNaC::_num5_p`
- `const ex GiNaC::_ex5 = ex(*_num5_p)`
- `const numeric * GiNaC::_num6_p`
- `const ex GiNaC::_ex6 = ex(*_num6_p)`
- `const numeric * GiNaC::_num7_p`
- `const ex GiNaC::_ex7 = ex(*_num7_p)`
- `const numeric * GiNaC::_num8_p`
- `const ex GiNaC::_ex8 = ex(*_num8_p)`
- `const numeric * GiNaC::_num9_p`
- `const ex GiNaC::_ex9 = ex(*_num9_p)`
- `const numeric * GiNaC::_num10_p`
- `const ex GiNaC::_ex10 = ex(*_num10_p)`
- `const numeric * GiNaC::_num11_p`
- `const ex GiNaC::_ex11 = ex(*_num11_p)`
- `const numeric * GiNaC::_num12_p`
- `const ex GiNaC::_ex12 = ex(*_num12_p)`
- `const numeric * GiNaC::_num15_p`
- `const ex GiNaC::_ex15 = ex(*_num15_p)`
- `const numeric * GiNaC::_num18_p`
- `const ex GiNaC::_ex18 = ex(*_num18_p)`
- `const numeric * GiNaC::_num20_p`
- `const ex GiNaC::_ex20 = ex(*_num20_p)`
- `const numeric * GiNaC::_num24_p`
- `const ex GiNaC::_ex24 = ex(*_num24_p)`
- `const numeric * GiNaC::_num25_p`
- `const ex GiNaC::_ex25 = ex(*_num25_p)`
- `const numeric * GiNaC::_num30_p`
- `const ex GiNaC::_ex30 = ex(*_num30_p)`
- `const numeric * GiNaC::_num48_p`
- `const ex GiNaC::_ex48 = ex(*_num48_p)`
- `const numeric * GiNaC::_num60_p`
- `const ex GiNaC::_ex60 = ex(*_num60_p)`
- `const numeric * GiNaC::_num120_p`
- `const ex GiNaC::_ex120 = ex(*_num120_p)`

### 9.92.1 Detailed Description

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 9.93 utils.h File Reference

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "assertion.h"
#include <functional>
#include <cstdlib>
#include <string>
```

### Classes

- class [GiNaC::dunno](#)  
*Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold().*
- class [GiNaC::basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.*
- struct [GiNaC::basic\\_partition\\_generator::mpartition2](#)
- class [GiNaC::partition\\_with\\_zero\\_parts\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.*
- class [GiNaC::partition\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.*
- class [GiNaC::composition\\_generator](#)  
*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*
- struct [GiNaC::composition\\_generator::coolmulti](#)
- struct [GiNaC::composition\\_generator::coolmulti::element](#)

### Namespaces

- namespace [GiNaC](#)

### Macros

- #define [DEFAULT\\_CTOR](#)(classname)
- #define [DEFAULT\\_COMPARE](#)(classname)
- #define [DEFAULT\\_PRINT](#)(classname, text)
- #define [DEFAULT\\_PRINT\\_LATEX](#)(classname, text, latex)

## Functions

- unsigned [GiNaC::log2](#) (unsigned n)  
*Integer binary logarithm.*
- unsigned [GiNaC::rotate\\_left](#) (unsigned n)  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T>  
int [GiNaC::compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [GiNaC::golden\\_ratio\\_hash](#) (uintptr\_t n)  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It>  
int [GiNaC::permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap>  
int [GiNaC::permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Cmp, class Swap>  
void [GiNaC::shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Swap>  
void [GiNaC::cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient n!*

### 9.93.1 Detailed Description

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

### 9.93.2 Macro Definition Documentation

#### 9.93.2.1 DEFAULT\_CTOR

```
#define DEFAULT_CTOR(  
    classname)
```

##### Value:

```
classname::classname() { setflag(status_flags::evaluated | status_flags::expanded); }
```

#### 9.93.2.2 DEFAULT\_COMPARE

```
#define DEFAULT_COMPARE(  
    classname)
```

##### Value:

```
int classname::compare_same_type(const basic & other) const \
{ \
    /* by default, the objects are always identical */ \
    return 0; \
}
```

### 9.93.2.3 DEFAULT\_PRINT

```
#define DEFAULT_PRINT(
    classname,
    text)
```

#### Value:

```
void classname::do_print(const print_context & c, unsigned level) const \
{ \
    c.s « text; \
}
```

### 9.93.2.4 DEFAULT\_PRINT\_LATEX

```
#define DEFAULT_PRINT_LATEX(
    classname,
    text,
    latex)
```

#### Value:

```
DEFAULT_PRINT(classname, text) \
void classname::do_print_latex(const print_latex & c, unsigned level) const \
{ \
    c.s « latex; \
}
```

## 9.94 utils\_multi\_iterator.h File Reference

Utilities for summing over multiple indices.

```
#include <cstdint>
#include <vector>
#include <ostream>
#include <iterator>
```

### Classes

- class [GiNaC::has\\_distance< T >](#)  
*SFINAE test for distance.*
- class [GiNaC::basic\\_multi\\_iterator< T >](#)  
*basic\_multi\_iterator is a base class.*
- class [GiNaC::multi\\_iterator\\_ordered< T >](#)  
*The class multi\_iterator\_ordered defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#)  
*The class multi\_iterator\_ordered\_eq defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#)  
*The class multi\_iterator\_ordered\_eq\_indv defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_counter< T >](#)  
*The class multi\_iterator\_counter defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#)  
*The class multi\_iterator\_counter\_indv defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_permutation< T >](#)  
*The class multi\_iterator\_permutation defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.*
- class [GiNaC::multi\\_iterator\\_shuffle< T >](#)  
*The class multi\_iterator\_shuffle defines a multi\_iterator, which runs over all shuffles of a and b.*
- class [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#)  
*The class multi\_iterator\_shuffle\_prime defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `template<typename T>`  
`std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`GiNaC::format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T>`  
`std::enable_if<!has\_distance< T >::value, T >::type GiNaC::format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const basic\_multi\_iterator< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_counter< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_counter\_indv< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_permutation< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_shuffle< T > &v)`  
*Output operator.*
- `template<class T>`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_shuffle\_prime< T > &v)`  
*Output operator.*

### 9.94.1 Detailed Description

Utilities for summing over multiple indices.

## 9.95 version.h File Reference

[GiNaC](#) library version information.

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define` [GINACLIB\\_MAJOR\\_VERSION](#) 1
- `#define` [GINACLIB\\_MINOR\\_VERSION](#) 8
- `#define` [GINACLIB\\_MICRO\\_VERSION](#) 10
- `#define` [GINAC\\_LT\\_CURRENT](#) 14
- `#define` [GINAC\\_LT\\_REVISION](#) 2
- `#define` [GINAC\\_LT\\_AGE](#) 1
- `#define` [GINACLIB\\_ARCHIVE\\_VERSION](#) 3
- `#define` [GINACLIB\\_ARCHIVE\\_AGE](#) 3
- `#define` [GINACLIB\\_STR\\_HELPER](#)(x)
- `#define` [GINACLIB\\_STR](#)(x)
- `#define` [GINACLIB\\_VERSION](#)

### 9.95.1 Detailed Description

[GiNaC](#) library version information.

### 9.95.2 Macro Definition Documentation

#### 9.95.2.1 GINACLIB\_MAJOR\_VERSION

```
#define GINACLIB_MAJOR_VERSION 1
```

#### 9.95.2.2 GINACLIB\_MINOR\_VERSION

```
#define GINACLIB_MINOR_VERSION 8
```

#### 9.95.2.3 GINACLIB\_MICRO\_VERSION

```
#define GINACLIB_MICRO_VERSION 10
```

#### 9.95.2.4 GINAC\_LT\_CURRENT

```
#define GINAC_LT_CURRENT 14
```

#### 9.95.2.5 GINAC\_LT\_REVISION

```
#define GINAC_LT_REVISION 2
```

### 9.95.2.6 GINAC\_LT\_AGE

```
#define GINAC_LT_AGE 1
```

### 9.95.2.7 GINACLIB\_ARCHIVE\_VERSION

```
#define GINACLIB_ARCHIVE_VERSION 3
```

Referenced by [GiNaC::archive::operator<<](#), and [GiNaC::archive::operator>>](#).

### 9.95.2.8 GINACLIB\_ARCHIVE\_AGE

```
#define GINACLIB_ARCHIVE_AGE 3
```

Referenced by [GiNaC::archive::operator>>](#).

### 9.95.2.9 GINACLIB\_STR\_HELPER

```
#define GINACLIB_STR_HELPER(  
    x)
```

#### Value:

```
#x
```

### 9.95.2.10 GINACLIB\_STR

```
#define GINACLIB_STR(  
    x)
```

#### Value:

```
GINACLIB\_STR\_HELPER(x)
```

### 9.95.2.11 GINACLIB\_VERSION

```
#define GINACLIB_VERSION
```

#### Value:

```
GINACLIB\_STR(GINACLIB_MAJOR_VERSION) " " \  
GINACLIB\_STR(GINACLIB_MINOR_VERSION) " " \  
GINACLIB\_STR(GINACLIB_MICRO_VERSION)
```

## 9.96 wildcard.cpp File Reference

Implementation of [GiNaC](#)'s wildcard objects.

```
#include "wildcard.h"  
#include "archive.h"  
#include "utils.h"  
#include "hash_seed.h"
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([wildcard](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[wildcard::do\\_print](#)), [print\\_func](#)< [print\\_tree](#)>(&[wildcard::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#)>(&[wildcard::do\\_print\\_python\\_repr](#))) [wildcard](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([wildcard](#))
- [bool GiNaC::haswild](#) (const [ex](#) &x)

*Check whether x has a wildcard anywhere as a subexpression.*

### 9.96.1 Detailed Description

Implementation of [GiNaC](#)'s wildcard objects.

## 9.97 wildcard.h File Reference

Interface to [GiNaC](#)'s wildcard objects.

```
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::wildcard](#)  
*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([wildcard](#))
- [ex GiNaC::wild](#) (unsigned label=0)  
*Create a wildcard object with the specified label.*
- [bool GiNaC::haswild](#) (const [ex](#) &x)  
*Check whether x has a wildcard anywhere as a subexpression.*

### 9.97.1 Detailed Description

Interface to [GiNaC](#)'s wildcard objects.



# Index

[\\_ex0](#)  
    [GiNaC, 292](#)

[\\_ex1](#)  
    [GiNaC, 294](#)

[\\_ex10](#)  
    [GiNaC, 297](#)

[\\_ex11](#)  
    [GiNaC, 297](#)

[\\_ex12](#)  
    [GiNaC, 298](#)

[\\_ex120](#)  
    [GiNaC, 300](#)

[\\_ex15](#)  
    [GiNaC, 298](#)

[\\_ex18](#)  
    [GiNaC, 298](#)

[\\_ex1\\_2](#)  
    [GiNaC, 294](#)

[\\_ex1\\_3](#)  
    [GiNaC, 293](#)

[\\_ex1\\_4](#)  
    [GiNaC, 293](#)

[\\_ex2](#)  
    [GiNaC, 295](#)

[\\_ex20](#)  
    [GiNaC, 299](#)

[\\_ex24](#)  
    [GiNaC, 299](#)

[\\_ex25](#)  
    [GiNaC, 299](#)

[\\_ex3](#)  
    [GiNaC, 295](#)

[\\_ex30](#)  
    [GiNaC, 299](#)

[\\_ex4](#)  
    [GiNaC, 295](#)

[\\_ex48](#)  
    [GiNaC, 300](#)

[\\_ex5](#)  
    [GiNaC, 296](#)

[\\_ex6](#)  
    [GiNaC, 296](#)

[\\_ex60](#)  
    [GiNaC, 300](#)

[\\_ex7](#)  
    [GiNaC, 296](#)

[\\_ex8](#)  
    [GiNaC, 297](#)

[\\_ex9](#)  
    [GiNaC, 297](#)

[\\_ex\\_1](#)  
    [GiNaC, 291](#)

[\\_ex\\_10](#)  
    [GiNaC, 289](#)

[\\_ex\\_11](#)  
    [GiNaC, 289](#)

[\\_ex\\_12](#)  
    [GiNaC, 288](#)

[\\_ex\\_120](#)  
    [GiNaC, 286](#)

[\\_ex\\_15](#)  
    [GiNaC, 288](#)

[\\_ex\\_18](#)  
    [GiNaC, 288](#)

[\\_ex\\_1\\_2](#)  
    [GiNaC, 292](#)

[\\_ex\\_1\\_3](#)  
    [GiNaC, 292](#)

[\\_ex\\_1\\_4](#)  
    [GiNaC, 292](#)

[\\_ex\\_2](#)  
    [GiNaC, 291](#)

[\\_ex\\_20](#)  
    [GiNaC, 288](#)

[\\_ex\\_24](#)  
    [GiNaC, 287](#)

[\\_ex\\_25](#)  
    [GiNaC, 287](#)

[\\_ex\\_3](#)  
    [GiNaC, 291](#)

[\\_ex\\_30](#)  
    [GiNaC, 287](#)

[\\_ex\\_4](#)  
    [GiNaC, 290](#)

[\\_ex\\_48](#)  
    [GiNaC, 287](#)

[\\_ex\\_5](#)  
    [GiNaC, 290](#)

[\\_ex\\_6](#)  
    [GiNaC, 290](#)

[\\_ex\\_60](#)  
    [GiNaC, 286](#)

[\\_ex\\_7](#)  
    [GiNaC, 290](#)

[\\_ex\\_8](#)  
    [GiNaC, 289](#)

[\\_ex\\_9](#)  
    [GiNaC, 289](#)

- `_iter_rep`
  - `GiNaC::internal::_iter_rep`, 306
- `_num0_bp`
  - `GiNaC`, 284
- `_num0_p`
  - `GiNaC`, 292
- `_num10_p`
  - `GiNaC`, 297
- `_num11_p`
  - `GiNaC`, 297
- `_num120_p`
  - `GiNaC`, 300
- `_num12_p`
  - `GiNaC`, 298
- `_num15_p`
  - `GiNaC`, 298
- `_num18_p`
  - `GiNaC`, 298
- `_num1_2_p`
  - `GiNaC`, 293
- `_num1_3_p`
  - `GiNaC`, 293
- `_num1_4_p`
  - `GiNaC`, 293
- `_num1_p`
  - `GiNaC`, 294
- `_num20_p`
  - `GiNaC`, 298
- `_num24_p`
  - `GiNaC`, 299
- `_num25_p`
  - `GiNaC`, 299
- `_num2_p`
  - `GiNaC`, 295
- `_num30_p`
  - `GiNaC`, 299
- `_num3_p`
  - `GiNaC`, 295
- `_num48_p`
  - `GiNaC`, 299
- `_num4_p`
  - `GiNaC`, 295
- `_num5_p`
  - `GiNaC`, 296
- `_num60_p`
  - `GiNaC`, 300
- `_num6_p`
  - `GiNaC`, 296
- `_num7_p`
  - `GiNaC`, 296
- `_num8_p`
  - `GiNaC`, 296
- `_num9_p`
  - `GiNaC`, 297
- `_num_10_p`
  - `GiNaC`, 289
- `_num_11_p`
  - `GiNaC`, 288
- `_num_120_p`
  - `GiNaC`, 286
- `_num_12_p`
  - `GiNaC`, 288
- `_num_15_p`
  - `GiNaC`, 288
- `_num_18_p`
  - `GiNaC`, 288
- `_num_1_2_p`
  - `GiNaC`, 291
- `_num_1_3_p`
  - `GiNaC`, 292
- `_num_1_4_p`
  - `GiNaC`, 292
- `_num_1_p`
  - `GiNaC`, 291
- `_num_20_p`
  - `GiNaC`, 287
- `_num_24_p`
  - `GiNaC`, 287
- `_num_25_p`
  - `GiNaC`, 287
- `_num_2_p`
  - `GiNaC`, 291
- `_num_30_p`
  - `GiNaC`, 287
- `_num_3_p`
  - `GiNaC`, 290
- `_num_48_p`
  - `GiNaC`, 286
- `_num_4_p`
  - `GiNaC`, 290
- `_num_5_p`
  - `GiNaC`, 290
- `_num_60_p`
  - `GiNaC`, 286
- `_num_6_p`
  - `GiNaC`, 290
- `_num_7_p`
  - `GiNaC`, 289
- `_num_8_p`
  - `GiNaC`, 289
- `_num_9_p`
  - `GiNaC`, 289
- `_numeric_digits`
  - `GiNaC::_numeric_digits`, 308
- `~archive`
  - `GiNaC::archive`, 328
- `~basic`
  - `GiNaC::basic`, 351
- `~basic_multi_iterator`
  - `GiNaC::basic_multi_iterator< T >`, 384
- `~compare_all_equal`
  - `GiNaC::compare_all_equal< T >`, 425
- `~compare_bitwise`
  - `GiNaC::compare_bitwise< T >`, 426
- `~compare_std_less`
  - `GiNaC::compare_std_less< T >`, 427

- ~container\_storage
  - GiNaC::container\_storage< C >, 471
- ~coolmulti
  - GiNaC::composition\_generator::coolmulti, 473
- ~element
  - GiNaC::composition\_generator::coolmulti::element, 534
- ~function\_options
  - GiNaC::function\_options, 657
- ~library\_init
  - GiNaC::library\_init, 794
- ~map\_function
  - GiNaC::map\_function, 799
- ~print\_context
  - GiNaC::print\_context, 999
- ~print\_functor\_impl
  - GiNaC::print\_functor\_impl, 1014
- ~ptr
  - GiNaC::ptr< T >, 1050
- ~unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 1218
- ~visitor
  - GiNaC::visitor, 1236
- a
  - GiNaC::archive\_node, 343
  - GiNaC::Eisenstein\_kernel, 533
  - GiNaC::integral, 758
- abs
  - GiNaC, 240
- abs\_conjugate
  - GiNaC, 147
- abs\_eval
  - GiNaC, 146
- abs\_evalf
  - GiNaC, 146
- abs\_expand
  - GiNaC, 146
- abs\_expl\_derivative
  - GiNaC, 146
- abs\_imag\_part
  - GiNaC, 147
- abs\_info
  - GiNaC, 148
- abs\_power
  - GiNaC, 147
- abs\_print\_csrc\_float
  - GiNaC, 147
- abs\_print\_latex
  - GiNaC, 146
- abs\_real\_part
  - GiNaC, 147
- accept
  - GiNaC::basic, 358
  - GiNaC::ex, 564
- access\_counter
  - GiNaC::remember\_table\_entry, 1084
- acos
  - GiNaC, 232
- acos\_conjugate
  - GiNaC, 188
- acos\_deriv
  - GiNaC, 187
- acos\_eval
  - GiNaC, 187
- acos\_evalf
  - GiNaC, 187
- acosh
  - GiNaC, 235
- acosh\_conjugate
  - GiNaC, 196
- acosh\_deriv
  - GiNaC, 196
- acosh\_eval
  - GiNaC, 196
- acosh\_evalf
  - GiNaC, 196
- adaptivesimpson
  - GiNaC, 198
- add
  - GiNaC::add, 316, 317
  - GiNaC::Gt\_detail::pathintegral\_term\_list, 951
  - GiNaC::matrix, 811
  - GiNaC::mul, 858
  - GiNaC::numeric, 926
  - GiNaC::scalar\_products, 1090
  - GiNaC::symmetry, 1180
- add.cpp, 1245
- add.h, 1246
- add\_bool
  - GiNaC::archive\_node, 337
- add\_callback
  - GiNaC::numeric\_digits, 308
- add\_child
  - GiNaC::class\_info< OPT >::tree\_node, 1217
- add\_dyn
  - GiNaC::numeric, 928
- add\_entry
  - GiNaC::remember\_table, 1080
  - GiNaC::remember\_table\_list, 1086
- add\_ex
  - GiNaC::archive\_node, 338
- add\_indexed
  - GiNaC::basic, 364
  - GiNaC::matrix, 808
  - GiNaC::structure< T, ComparisonPolicy >, 1128
- add\_node
  - GiNaC::archive, 331
- add\_reference
  - GiNaC::refcounted, 1062
- add\_series
  - GiNaC::pseries, 1043
- add\_string
  - GiNaC::archive\_node, 338
- add\_symbol
  - GiNaC, 214
- add\_unsigned

- GiNaC::archive\_node, [337](#)
- add\_vectors
  - GiNaC::scalar\_products, [1090](#)
- after\_i
  - GiNaC::composition\_generator::coolmulti, [474](#)
- algebraic
  - GiNaC::has\_options, [712](#)
  - GiNaC::subs\_options, [1154](#)
- algebraic\_match\_mul\_with\_mul
  - GiNaC, [211](#)
- algebraic\_subs\_mul
  - GiNaC::mul, [856](#)
- all
  - GiNaC::factor\_options, [611](#)
- all\_index\_values\_are
  - GiNaC::indexed, [742](#)
- antisymmetric
  - GiNaC::symmetry, [1178](#)
- antisymmetric2
  - GiNaC, [266](#)
- antisymmetric3
  - GiNaC, [266](#)
- antisymmetric4
  - GiNaC, [266](#)
- antisymmetrize
  - GiNaC, [122](#), [123](#), [267](#), [271](#)
  - GiNaC::ex, [576](#)
- append
  - GiNaC::container< C >, [466](#)
- append\_factors
  - GiNaC::ncmul, [909](#)
- apply\_function\_recursive
  - GiNaC::Gt, [707](#)
- applyIntegrationPath
  - GiNaC::Gt, [704](#)
- archive
  - GiNaC::archive, [328](#), [329](#)
  - GiNaC::basic, [368](#)
  - GiNaC::clifford, [402](#)
  - GiNaC::color, [422](#)
  - GiNaC::constant, [450](#)
  - GiNaC::container< C >, [463](#)
  - GiNaC::expairseq, [599](#)
  - GiNaC::fderivative, [626](#)
  - GiNaC::function, [645](#)
  - GiNaC::Gt, [702](#)
  - GiNaC::idx, [721](#)
  - GiNaC::indexed, [740](#)
  - GiNaC::integral, [756](#)
  - GiNaC::matrix, [809](#)
  - GiNaC::minkmetric, [829](#)
  - GiNaC::numeric, [925](#)
  - GiNaC::power, [993](#)
  - GiNaC::pseries, [1040](#)
  - GiNaC::relational, [1073](#)
  - GiNaC::spinidx, [1101](#)
  - GiNaC::symbol, [1168](#)
  - GiNaC::symmetry, [1179](#)
  - GiNaC::tensepsilon, [1198](#)
  - GiNaC::varidx, [1233](#)
  - GiNaC::wildcard, [1241](#)
- archive.cpp, [1246](#)
- archive.h, [1247](#)
  - GINAC\_BIND\_UNARCHIVER, [1249](#)
  - GINAC\_DECLARE\_UNARCHIVER, [1248](#)
- archive\_atom
  - GiNaC, [68](#)
- archive\_ex
  - GiNaC::archive, [329](#)
- archive\_node
  - GiNaC::archive\_node, [337](#)
  - GiNaC::ex, [580](#)
- archive\_node\_cit
  - GiNaC::archive\_node, [336](#)
- archive\_node\_id
  - GiNaC, [68](#)
- archived\_ex
  - GiNaC::archive::archived\_ex, [345](#)
- are\_ex\_trivially\_equal
  - GiNaC, [116](#)
  - GiNaC::ex, [580](#)
- arg1
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [956](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [959](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [961](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [967](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [970](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [973](#)
- arg2
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [959](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [961](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [971](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [974](#)
- arg3
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [962](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [974](#)
- args
  - GiNaC::Gt, [709](#)
- argument\_type
  - GiNaC::map\_function, [799](#)
- asin
  - GiNaC, [232](#)
- asin\_conjugate
  - GiNaC, [186](#)

- asin\_deriv
  - GiNaC, [186](#)
- asin\_eval
  - GiNaC, [186](#)
- asin\_evalf
  - GiNaC, [186](#)
- asin\_info
  - GiNaC, [187](#)
- asinh
  - GiNaC, [234](#)
- asinh\_conjugate
  - GiNaC, [195](#)
- asinh\_deriv
  - GiNaC, [195](#)
- asinh\_eval
  - GiNaC, [195](#)
- asinh\_evalf
  - GiNaC, [195](#)
- assertion.h, [1249](#)
  - GINAC\_ASSERT, [1250](#)
- atan
  - GiNaC, [232](#), [233](#)
- atan2\_deriv
  - GiNaC, [190](#)
- atan2\_eval
  - GiNaC, [190](#)
- atan2\_evalf
  - GiNaC, [189](#)
- atan2\_info
  - GiNaC, [190](#)
- atan\_conjugate
  - GiNaC, [189](#)
- atan\_deriv
  - GiNaC, [188](#)
- atan\_eval
  - GiNaC, [188](#)
- atan\_evalf
  - GiNaC, [188](#)
- atan\_info
  - GiNaC, [189](#)
- atan\_series
  - GiNaC, [189](#)
- atanh
  - GiNaC, [235](#)
- atanh\_conjugate
  - GiNaC, [198](#)
- atanh\_deriv
  - GiNaC, [197](#)
- atanh\_eval
  - GiNaC, [197](#)
- atanh\_evalf
  - GiNaC, [197](#)
- atanh\_series
  - GiNaC, [197](#)
- atend
  - GiNaC::composition\_generator, [429](#)
- atomize
  - GiNaC::archive, [332](#)
- atoms
  - GiNaC::archive, [333](#)
- attribute\_pure
  - compiler.h, [1262](#)
- auto\_clear\_polylog\_cache
  - GiNaC::Gt, [709](#)
- automatic
  - GiNaC::determinant\_algo, [477](#)
  - GiNaC::solve\_algo, [1092](#)
- B
  - GiNaC::basic\_multi\_iterator< T >, [387](#)
- b
  - GiNaC::Eisenstein\_kernel, [533](#)
  - GiNaC::integral, [758](#)
- bareiss
  - GiNaC::determinant\_algo, [478](#)
  - GiNaC::solve\_algo, [1093](#)
- base\_and\_index
  - GiNaC, [98](#)
- basic
  - GiNaC::basic, [351](#)
- basic.cpp, [1251](#)
- basic.h, [1252](#)
- basic\_multi\_iterator
  - GiNaC::basic\_multi\_iterator< T >, [384](#)
- basic\_partition\_generator
  - GiNaC::basic\_partition\_generator, [390](#)
- basis
  - GiNaC::power, [998](#)
- begin
  - GiNaC::archive\_node::archive\_node\_cit\_range, [345](#)
  - GiNaC::container< C >, [467](#)
  - GiNaC::ex, [556](#)
  - GiNaC::Gt\_detail::pathintegral\_term\_list, [950](#)
- bernoulli
  - GiNaC, [239](#)
- Bernoulli\_polynomial
  - GiNaC, [201](#)
- beta\_deriv
  - GiNaC, [169](#)
- beta\_eval
  - GiNaC, [169](#)
- beta\_evalf
  - GiNaC, [169](#)
- beta\_series
  - GiNaC, [169](#)
- binomial
  - GiNaC, [239](#)
- binomial\_conjugate
  - GiNaC, [157](#)
- binomial\_eval
  - GiNaC, [156](#)
- binomial\_evalf
  - GiNaC, [156](#)
- binomial\_imag\_part
  - GiNaC, [157](#)
- binomial\_real\_part

- GiNaC, [157](#)
- binomial\_sym
  - GiNaC, [156](#)
- bp
  - GiNaC::ex, [582](#)
- c
  - GiNaC::pointer\_to\_member\_to\_map\_function< C  
>, [964](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg<  
C, T1 >, [967](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args<  
C, T1, T2 >, [970](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args<  
C, T1, T2, T3 >, [973](#)
- C\_norm
  - GiNaC::Eisenstein\_h\_kernel, [523](#)
  - GiNaC::Eisenstein\_kernel, [533](#)
  - GiNaC::Kronecker\_dtau\_kernel, [781](#)
  - GiNaC::Kronecker\_dz\_kernel, [790](#)
  - GiNaC::modular\_form\_kernel, [838](#)
- cache
  - GiNaC::Gt\_detail::TransformExpressionWithCache<  
Type >, [1216](#)
- cache\_step\_size
  - GiNaC::integration\_kernel, [768](#)
- calc\_lanczos\_A
  - GiNaC::lanczos\_coefs, [791](#)
- calchash
  - GiNaC::basic, [367](#)
  - GiNaC::constant, [451](#)
  - GiNaC::expairseq, [600](#)
  - GiNaC::function, [643](#)
  - GiNaC::Gt, [701](#)
  - GiNaC::idx, [722](#)
  - GiNaC::numeric, [926](#)
  - GiNaC::relational, [1075](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1131](#)
  - GiNaC::symbol, [1169](#)
  - GiNaC::symmetry, [1179](#)
  - GiNaC::wildcard, [1241](#)
- callback\_registered
  - GiNaC, [92](#)
- callbacklist
  - GiNaC::\_numeric\_digits, [309](#)
- can\_be\_further\_expanded
  - GiNaC::mul, [857](#)
- can\_make\_flat
  - GiNaC::expairseq, [603](#)
  - GiNaC::mul, [855](#)
- canonical
  - GiNaC::relational, [1074](#)
- canonicalize
  - GiNaC, [267](#)
  - GiNaC::expairseq, [605](#)
  - GiNaC::symmetry, [1182](#)
- canonicalize\_clifford
  - GiNaC, [103](#)
- Catalan
  - GiNaC, [283](#)
- CatalanEvalf
  - GiNaC, [245](#)
- charpoly
  - GiNaC, [209](#)
  - GiNaC::matrix, [815](#)
- check
  - GiNaC::Gt::kernel, [772](#)
- check\_type
  - GiNaC::Gt\_detail::TransformExpressionWithCache<  
Type >, [1215](#)
- children
  - GiNaC::class\_info< OPT >::tree\_node, [1217](#)
  - GiNaC::symmetry, [1183](#)
- cinteger
  - GiNaC::info\_flags, [747](#)
- cinteger\_polynomial
  - GiNaC::info\_flags, [747](#)
- class\_info
  - GiNaC::class\_info< OPT >, [392](#)
- class\_info< OPT >::first
  - GiNaC, [282](#)
- class\_info< OPT >::parents\_identified
  - GiNaC, [283](#)
- class\_info.h, [1253](#)
- clear
  - GiNaC::archive, [331](#)
  - GiNaC::Gt\_detail::pathintegral\_term\_list, [950](#)
  - GiNaC::scalar\_products, [1090](#)
- clear\_all\_entries
  - GiNaC::remember\_table, [1080](#)
- clear\_cache
  - GiNaC::Gt\_detail::TransformExpressionWithCache<  
Type >, [1215](#)
- clear\_polylog\_cache
  - GiNaC::Gt, [707](#)
- clearflag
  - GiNaC::basic, [372](#)
- clifford
  - GiNaC::clifford, [401](#), [402](#)
- clifford.cpp, [1254](#)
- clifford.h, [1256](#)
- clifford\_bar
  - GiNaC, [110](#)
- clifford\_inverse
  - GiNaC, [105](#)
- clifford\_max\_label
  - GiNaC, [104](#)
- clifford\_moebius\_map
  - GiNaC, [107](#), [108](#)
- clifford\_norm
  - GiNaC, [105](#)
- clifford\_prime
  - GiNaC, [104](#)
- clifford\_star
  - GiNaC, [110](#)
- clifford\_star\_bar
  - GiNaC, [103](#)

- clifford\_to\_lst
  - GiNaC, 107
- clifford\_unit
  - GiNaC, 99
- cmgen
  - GiNaC::composition\_generator, 429
- CMPINDICES
  - color.cpp, 1259
- coeff
  - GiNaC, 119
  - GiNaC::add, 318
  - GiNaC::basic, 359
  - GiNaC::ex, 565
  - GiNaC::expair, 588
  - GiNaC::mul, 848
  - GiNaC::ncmul, 906
  - GiNaC::numeric, 921
  - GiNaC::power, 989
  - GiNaC::pseries, 1037
  - GiNaC::structure< T, ComparisonPolicy >, 1124
  - GiNaC::symminfo, 1185
- coefficient\_a0
  - GiNaC::Eisenstein\_h\_kernel, 522
- coefficient\_an
  - GiNaC::Eisenstein\_h\_kernel, 522
- coeffop
  - GiNaC::pseries, 1042
- coeffs
  - GiNaC::lanczos\_coeffs, 792
- coerce
  - GiNaC, 229
- coerce< int, cln::cl\_I >
  - GiNaC, 229
- coerce< unsigned int, cln::cl\_I >
  - GiNaC, 229
- col
  - GiNaC::matrix, 822
- collect
  - GiNaC, 120
  - GiNaC::basic, 359
  - GiNaC::ex, 566
  - GiNaC::pseries, 1037
  - GiNaC::structure< T, ComparisonPolicy >, 1125
- collect\_common\_factors
  - GiNaC, 226
- collect\_symbols
  - GiNaC, 214
- color
  - GiNaC::color, 421
- color.cpp, 1258
  - CMPINDICES, 1259
  - TEST\_PERMUTATION, 1259
- color.h, 1260
- color\_d
  - GiNaC, 113
- color\_f
  - GiNaC, 112
- color\_h
  - GiNaC, 113
- color\_ONE
  - GiNaC, 111
- color\_T
  - GiNaC, 112
- color\_trace
  - GiNaC, 114, 115
- cols
  - GiNaC, 209
  - GiNaC::matrix, 810
- combine\_ex\_with\_coeff\_to\_pair
  - GiNaC::add, 324
  - GiNaC::expairseq, 602
  - GiNaC::mul, 854
- combine\_indices
  - GiNaC::make\_flat\_inserter, 796
- combine\_overall\_coeff
  - GiNaC::expairseq, 603
  - GiNaC::mul, 855
- combine\_pair\_with\_coeff\_to\_pair
  - GiNaC::add, 324
  - GiNaC::expairseq, 602
  - GiNaC::mul, 854
- combine\_same\_terms\_sorted\_seq
  - GiNaC::expairseq, 606
- commutative
  - GiNaC::return\_types, 1088
- commutator\_sign
  - GiNaC::clifford, 407
- compare
  - GiNaC::basic, 370
  - GiNaC::ex, 574
  - GiNaC::expair, 587
  - GiNaC::numeric, 931
- compare\_pointers
  - GiNaC, 277
- compare\_same\_type
  - GiNaC::basic, 367
- compile\_ex
  - GiNaC, 127, 128
- compiler.h, 1261
  - attribute\_pure, 1262
  - likely, 1262
  - unlikely, 1262
- complex
  - GiNaC::domain, 504
- composition
  - GiNaC::composition\_generator, 430
- composition\_generator
  - GiNaC::composition\_generator, 429
- conjugate
  - GiNaC, 117
  - GiNaC::add, 321
  - GiNaC::basic, 366
  - GiNaC::constant, 449
  - GiNaC::container< C >, 463
  - GiNaC::diracgamma5, 488
  - GiNaC::diracgammaL, 493

- GiNaC::diracgammaR, [498](#)
  - GiNaC::ex, [561](#)
  - GiNaC::expair, [587](#)
  - GiNaC::expairseq, [598](#)
  - GiNaC::function, [644](#)
  - GiNaC::integral, [756](#)
  - GiNaC::matrix, [809](#)
  - GiNaC::mul, [852](#)
  - GiNaC::ncmul, [907](#)
  - GiNaC::numeric, [925](#)
  - GiNaC::power, [993](#)
  - GiNaC::pseries, [1039](#)
  - GiNaC::realsymbol, [1059](#)
  - GiNaC::spinidx, [1101](#)
  - GiNaC::symbol, [1167](#)
- conjugate\_conjugate
  - GiNaC, [141](#)
- conjugate\_eval
  - GiNaC, [141](#)
- conjugate\_evalf
  - GiNaC, [141](#)
- conjugate\_expl\_derivative
  - GiNaC, [141](#)
- conjugate\_f
  - GiNaC::function\_options, [687](#)
- conjugate\_func
  - GiNaC::function\_options, [662–664](#), [680](#)
- conjugate\_funcp
  - GiNaC, [70](#)
- conjugate\_funcp\_1
  - GiNaC, [71](#)
- conjugate\_funcp\_10
  - GiNaC, [83](#)
- conjugate\_funcp\_11
  - GiNaC, [84](#)
- conjugate\_funcp\_12
  - GiNaC, [85](#)
- conjugate\_funcp\_13
  - GiNaC, [87](#)
- conjugate\_funcp\_14
  - GiNaC, [88](#)
- conjugate\_funcp\_2
  - GiNaC, [72](#)
- conjugate\_funcp\_3
  - GiNaC, [74](#)
- conjugate\_funcp\_4
  - GiNaC, [75](#)
- conjugate\_funcp\_5
  - GiNaC, [76](#)
- conjugate\_funcp\_6
  - GiNaC, [77](#)
- conjugate\_funcp\_7
  - GiNaC, [79](#)
- conjugate\_funcp\_8
  - GiNaC, [80](#)
- conjugate\_funcp\_9
  - GiNaC, [81](#)
- conjugate\_funcp\_exvector
  - GiNaC, [90](#)
- conjugate\_imag\_part
  - GiNaC, [142](#)
- conjugate\_info
  - GiNaC, [142](#)
- conjugate\_print\_latex
  - GiNaC, [141](#)
- conjugate\_real\_part
  - GiNaC, [142](#)
- conjugate\_use\_exvector\_args
  - GiNaC::function\_options, [690](#)
- conjugateepvector
  - GiNaC, [129](#)
- const\_iterator
  - GiNaC::const\_iterator, [433](#)
  - GiNaC::container< C >, [460](#)
- const\_postorder\_iterator
  - GiNaC::const\_iterator, [436](#)
  - GiNaC::const\_postorder\_iterator, [438](#)
- const\_preorder\_iterator
  - GiNaC::const\_iterator, [435](#)
  - GiNaC::const\_preorder\_iterator, [441](#)
- const\_reverse\_iterator
  - GiNaC::container< C >, [460](#)
- constant
  - GiNaC::constant, [448](#)
- constant.cpp, [1262](#)
- constant.h, [1263](#)
- construct\_from\_2\_ex
  - GiNaC::expairseq, [604](#)
- construct\_from\_2\_expairseq
  - GiNaC::expairseq, [604](#)
- construct\_from\_basic
  - GiNaC::ex, [577](#)
- construct\_from\_double
  - GiNaC::ex, [579](#)
- construct\_from\_epvector
  - GiNaC::expairseq, [605](#)
- construct\_from\_expairseq\_ex
  - GiNaC::expairseq, [604](#)
- construct\_from\_exvector
  - GiNaC::expairseq, [604](#)
- construct\_from\_int
  - GiNaC::ex, [577](#)
- construct\_from\_long
  - GiNaC::ex, [578](#)
- construct\_from\_longlong
  - GiNaC::ex, [578](#)
- construct\_from\_string\_and\_lst
  - GiNaC::ex, [579](#)
- construct\_from\_uint
  - GiNaC::ex, [578](#)
- construct\_from\_ulong
  - GiNaC::ex, [578](#)
- construct\_from\_ulonglong
  - GiNaC::ex, [579](#)
- container
  - GiNaC::container< C >, [460](#)



- container.h, [1264](#)
- container\_storage
  - GiNaC::container\_storage< C >, [470](#), [471](#)
- content
  - GiNaC::ex, [570](#)
- contract\_with
  - GiNaC::basic, [365](#)
  - GiNaC::cliffordunit, [413](#)
  - GiNaC::diracgamma, [483](#)
  - GiNaC::matrix, [808](#)
  - GiNaC::spinmetric, [1109](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1130](#)
  - GiNaC::su3d, [1137](#)
  - GiNaC::su3f, [1143](#)
  - GiNaC::su3t, [1153](#)
  - GiNaC::tensdelta, [1192](#)
  - GiNaC::tensepsilon, [1198](#)
  - GiNaC::tensmetric, [1205](#)
- convert\_H\_to\_Li
  - GiNaC, [162](#)
- convert\_to\_poly
  - GiNaC::pseries, [1042](#)
- coolmulti
  - GiNaC::composition\_generator::coolmulti, [473](#)
- cos
  - GiNaC, [231](#)
- cos\_conjugate
  - GiNaC, [184](#)
- cos\_deriv
  - GiNaC, [183](#)
- cos\_eval
  - GiNaC, [183](#)
- cos\_evalf
  - GiNaC, [183](#)
- cos\_imag\_part
  - GiNaC, [184](#)
- cos\_real\_part
  - GiNaC, [183](#)
- cosh
  - GiNaC, [234](#)
- cosh\_conjugate
  - GiNaC, [193](#)
- cosh\_deriv
  - GiNaC, [192](#)
- cosh\_eval
  - GiNaC, [192](#)
- cosh\_evalf
  - GiNaC, [192](#)
- cosh\_imag\_part
  - GiNaC, [193](#)
- cosh\_real\_part
  - GiNaC, [192](#)
- count
  - GiNaC::archive\_node::property\_info, [1030](#)
  - GiNaC::library\_init, [795](#)
- count\_dummy\_indices
  - GiNaC, [136](#)
- count\_factors
  - GiNaC::ncmul, [909](#)
- count\_free\_indices
  - GiNaC, [136](#)
- covariant
  - GiNaC::varidx, [1235](#)
- crational
  - GiNaC::info\_flags, [747](#)
- crational\_polynomial
  - GiNaC::info\_flags, [747](#)
- crc32
  - GiNaC, [116](#)
- crc32.h, [1264](#)
- crctab
  - GiNaC, [284](#)
- csgn
  - GiNaC, [246](#)
  - GiNaC::numeric, [930](#)
- csgn\_conjugate
  - GiNaC, [150](#)
- csgn\_eval
  - GiNaC, [150](#)
- csgn\_evalf
  - GiNaC, [149](#)
- csgn\_imag\_part
  - GiNaC, [150](#)
- csgn\_power
  - GiNaC, [151](#)
- csgn\_real\_part
  - GiNaC, [150](#)
- csgn\_series
  - GiNaC, [150](#)
- csrc
  - GiNaC, [258](#)
- csrc\_cl\_N
  - GiNaC, [259](#)
- csrc\_double
  - GiNaC, [259](#)
- csrc\_float
  - GiNaC, [258](#)
- current\_serial
  - GiNaC::function, [649](#)
- current\_updated
  - GiNaC::composition\_generator, [430](#)
  - GiNaC::partition\_generator, [943](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [945](#)
- current\_vector
  - GiNaC::lanczos\_coeffs, [792](#)
- cutIntegrationPath\_threshold\_imag
  - GiNaC::Gt, [709](#)
- cutIntegrationPath\_threshold\_real
  - GiNaC::Gt, [709](#)
- cyclic
  - GiNaC::symmetry, [1178](#)
- cyclic\_permutation
  - GiNaC, [278](#)
- dbgprint
  - GiNaC::basic, [354](#)
  - GiNaC::ex, [558](#)

- dbgprinttree
  - GiNaC::basic, 354
  - GiNaC::ex, 558
- DCOUT
  - factor.cpp, 1274
- DCOUT2
  - factor.cpp, 1274
- DCOUTVAR
  - factor.cpp, 1274
- DEBUG\_GT
  - Gt.cpp, 1292
- debugprint
  - GiNaC::scalar\_products, 1091
  - GiNaC::spmapkey, 1112
- DECLARE\_FUNCTION\_10P
  - function.h, 1288
- DECLARE\_FUNCTION\_11P
  - function.h, 1288
- DECLARE\_FUNCTION\_12P
  - function.h, 1288
- DECLARE\_FUNCTION\_13P
  - function.h, 1289
- DECLARE\_FUNCTION\_14P
  - function.h, 1289
- DECLARE\_FUNCTION\_1P
  - function.h, 1286
- DECLARE\_FUNCTION\_2P
  - function.h, 1286
- DECLARE\_FUNCTION\_3P
  - function.h, 1286
- DECLARE\_FUNCTION\_4P
  - function.h, 1286
- DECLARE\_FUNCTION\_5P
  - function.h, 1287
- DECLARE\_FUNCTION\_6P
  - function.h, 1287
- DECLARE\_FUNCTION\_7P
  - function.h, 1287
- DECLARE\_FUNCTION\_8P
  - function.h, 1287
- DECLARE\_FUNCTION\_9P
  - function.h, 1288
- decomp\_rational
  - GiNaC, 216
- decomposeIntegrationPath
  - GiNaC::Gt, 703
- deconcatenate\_path
  - GiNaC::Gt\_detail, 301
- deconcatenate\_path< ex >
  - GiNaC::Gt\_detail, 301
- deconcatenate\_path< Gt::kernel >
  - GiNaC::Gt\_detail, 301
- DEFAULT\_COMPARE
  - utils.h, 1359
- DEFAULT\_CTOR
  - utils.h, 1359
- default\_overall\_coeff
  - GiNaC::expairseq, 603
- GiNaC::mul, 855
- DEFAULT\_PRINT
  - utils.h, 1359
- DEFAULT\_PRINT\_LATEX
  - utils.h, 1360
- deform
  - GiNaC::Gt::kernel, 773
- deg
  - GiNaC::pole\_error, 976
- deg\_a
  - GiNaC::sym\_desc, 1159
- deg\_b
  - GiNaC::sym\_desc, 1159
- degree
  - GiNaC, 118
  - GiNaC::add, 318
  - GiNaC::basic, 358
  - GiNaC::ex, 565
  - GiNaC::integral, 754
  - GiNaC::mul, 848
  - GiNaC::ncmul, 905
  - GiNaC::numeric, 921
  - GiNaC::pole\_error, 975
  - GiNaC::power, 989
  - GiNaC::pseries, 1036
  - GiNaC::structure< T, ComparisonPolicy >, 1124
- delete\_cyclic
  - GiNaC::remember\_strategies, 1078
- delete\_lfu
  - GiNaC::remember\_strategies, 1078
- delete\_lru
  - GiNaC::remember\_strategies, 1078
- delete\_never
  - GiNaC::remember\_strategies, 1077
- delta\_indent
  - GiNaC::print\_tree, 1027
- delta\_tensor
  - GiNaC, 272
- denom
  - GiNaC, 119, 249
  - GiNaC::ex, 569
  - GiNaC::Gt\_detail::pathintegral\_term, 949
  - GiNaC::numeric, 937
- derivative
  - GiNaC::add, 322
  - GiNaC::basic, 361
  - GiNaC::constant, 450
  - GiNaC::fderivative, 627
  - GiNaC::function, 645
  - GiNaC::idx, 722
  - GiNaC::indexed, 741
  - GiNaC::integral, 757
  - GiNaC::mul, 852
  - GiNaC::ncmul, 908
  - GiNaC::numeric, 926
  - GiNaC::power, 994
  - GiNaC::pseries, 1041
  - GiNaC::structure< T, ComparisonPolicy >, 1125

- GiNaC::symbol, [1168](#)
- derivative\_f
  - GiNaC::function\_options, [688](#)
- derivative\_func
  - GiNaC::function\_options, [670–672](#), [681](#)
- derivative\_funcp
  - GiNaC, [70](#)
- derivative\_funcp\_1
  - GiNaC, [72](#)
- derivative\_funcp\_10
  - GiNaC, [83](#)
- derivative\_funcp\_11
  - GiNaC, [84](#)
- derivative\_funcp\_12
  - GiNaC, [86](#)
- derivative\_funcp\_13
  - GiNaC, [87](#)
- derivative\_funcp\_14
  - GiNaC, [89](#)
- derivative\_funcp\_2
  - GiNaC, [73](#)
- derivative\_funcp\_3
  - GiNaC, [74](#)
- derivative\_funcp\_4
  - GiNaC, [75](#)
- derivative\_funcp\_5
  - GiNaC, [76](#)
- derivative\_funcp\_6
  - GiNaC, [78](#)
- derivative\_funcp\_7
  - GiNaC, [79](#)
- derivative\_funcp\_8
  - GiNaC, [80](#)
- derivative\_funcp\_9
  - GiNaC, [82](#)
- derivative\_funcp\_exvector
  - GiNaC, [90](#)
- derivative\_map\_function
  - GiNaC::derivative\_map\_function, [476](#)
- derivative\_use\_exvector\_args
  - GiNaC::function\_options, [691](#)
- derivatives
  - GiNaC::fderivative, [628](#)
- descend
  - GiNaC::const\_postorder\_iterator, [439](#)
- determinant
  - GiNaC, [209](#)
  - GiNaC::matrix, [814](#)
- determinant\_minor
  - GiNaC::matrix, [818](#)
- dfft
  - GiNaC, [258](#)
- diag\_matrix
  - GiNaC, [206](#), [207](#)
- diff
  - GiNaC, [121](#)
  - GiNaC::basic, [370](#)
  - GiNaC::ex, [566](#)
- difference\_type
  - GiNaC::const\_iterator, [432](#)
  - GiNaC::const\_postorder\_iterator, [437](#)
  - GiNaC::const\_preorder\_iterator, [440](#)
- Digits
  - GiNaC, [285](#)
- digits
  - GiNaC::numeric\_digits, [309](#)
- digits\_changed\_callback
  - GiNaC, [92](#)
- dim
  - GiNaC::idx, [726](#)
  - GiNaC::spmapkey, [1113](#)
- dirac\_gamma
  - GiNaC, [100](#)
- dirac\_gamma5
  - GiNaC, [100](#)
- dirac\_gammaL
  - GiNaC, [100](#)
- dirac\_gammaR
  - GiNaC, [101](#)
- dirac\_ONE
  - GiNaC, [99](#)
- dirac\_slash
  - GiNaC, [101](#)
- dirac\_trace
  - GiNaC, [102](#), [103](#)
- dirichlet\_character
  - GiNaC, [200](#)
- div
  - GiNaC::numeric, [927](#)
- div\_dyn
  - GiNaC::numeric, [928](#)
- divfree
  - GiNaC::determinant\_algo, [477](#)
  - GiNaC::solve\_algo, [1093](#)
- divide
  - GiNaC, [218](#)
- divide\_in\_z
  - GiNaC, [218](#)
- division\_free\_elimination
  - GiNaC::matrix, [819](#)
- do\_not\_evalf\_params
  - GiNaC::function\_options, [684](#)
- do\_print
  - GiNaC::add, [325](#)
  - GiNaC::basic, [373](#)
  - GiNaC::basic\_log\_kernel, [381](#)
  - GiNaC::cliffordunit, [413](#)
  - GiNaC::constant, [452](#)
  - GiNaC::container< C >, [468](#)
  - GiNaC::diracgamma, [483](#)
  - GiNaC::diracgamma5, [488](#)
  - GiNaC::diracgammaL, [493](#)
  - GiNaC::diracgammaR, [498](#)
  - GiNaC::diracone, [503](#)
  - GiNaC::Ebar\_kernel, [512](#)
  - GiNaC::Eisenstein\_h\_kernel, [522](#)

- GiNaC::Eisenstein\_kernel, 532
- GiNaC::ELi\_kernel, 542
- GiNaC::expairseq, 603
- GiNaC::fail, 616
- GiNaC::fderivative, 628
- GiNaC::Gt, 708
- GiNaC::idx, 725
- GiNaC::indexed, 744
- GiNaC::integral, 757
- GiNaC::integration\_kernel, 767
- GiNaC::Kronecker\_dtau\_kernel, 780
- GiNaC::Kronecker\_dz\_kernel, 789
- GiNaC::matrix, 821
- GiNaC::minkmetric, 829
- GiNaC::modular\_form\_kernel, 838
- GiNaC::mul, 856
- GiNaC::multiple\_polylog\_kernel, 897
- GiNaC::ncmul, 909
- GiNaC::numeric, 938
- GiNaC::pseries, 1045
- GiNaC::relational, 1075
- GiNaC::spinidx, 1103
- GiNaC::spinmetric, 1110
- GiNaC::su3d, 1138
- GiNaC::su3f, 1143
- GiNaC::su3one, 1148
- GiNaC::su3t, 1153
- GiNaC::symbol, 1170
- GiNaC::symmetry, 1181
- GiNaC::tensdelta, 1192
- GiNaC::tensepsilon, 1199
- GiNaC::tensmetric, 1205
- GiNaC::user\_defined\_kernel, 1226
- GiNaC::varidx, 1235
- GiNaC::wildcard, 1242
- do\_print\_csrc
  - GiNaC::add, 325
  - GiNaC::fderivative, 629
  - GiNaC::idx, 725
  - GiNaC::mul, 857
  - GiNaC::ncmul, 909
  - GiNaC::numeric, 939
  - GiNaC::power, 996
- do\_print\_csrc\_cl\_N
  - GiNaC::numeric, 939
  - GiNaC::power, 996
- do\_print\_dflt
  - GiNaC::clifford, 406
  - GiNaC::power, 995
- do\_print\_latex
  - GiNaC::add, 325
  - GiNaC::clifford, 407
  - GiNaC::cliffordunit, 413
  - GiNaC::constant, 452
  - GiNaC::diracgamma, 483
  - GiNaC::diracgamma5, 488
  - GiNaC::diracgammaL, 493
  - GiNaC::diracgammaR, 498
- GiNaC::diracone, 503
- GiNaC::fderivative, 628
- GiNaC::Gt, 708
- GiNaC::idx, 725
- GiNaC::indexed, 744
- GiNaC::integral, 758
- GiNaC::matrix, 821
- GiNaC::minkmetric, 829
- GiNaC::mul, 857
- GiNaC::numeric, 938
- GiNaC::power, 995
- GiNaC::pseries, 1045
- GiNaC::spinidx, 1103
- GiNaC::spinmetric, 1110
- GiNaC::su3d, 1138
- GiNaC::su3f, 1143
- GiNaC::su3one, 1148
- GiNaC::su3t, 1153
- GiNaC::symbol, 1170
- GiNaC::tensdelta, 1192
- GiNaC::tensepsilon, 1199
- do\_print\_python
  - GiNaC::container< C >, 468
  - GiNaC::power, 996
  - GiNaC::pseries, 1045
- do\_print\_python\_repr
  - GiNaC::add, 326
  - GiNaC::basic, 373
  - GiNaC::constant, 452
  - GiNaC::container< C >, 468
  - GiNaC::matrix, 821
  - GiNaC::mul, 857
  - GiNaC::numeric, 939
  - GiNaC::power, 996
  - GiNaC::pseries, 1046
  - GiNaC::relational, 1075
  - GiNaC::symbol, 1170
  - GiNaC::wildcard, 1242
- do\_print\_tree
  - GiNaC::basic, 373
  - GiNaC::clifford, 407
  - GiNaC::constant, 452
  - GiNaC::container< C >, 468
  - GiNaC::expairseq, 604
  - GiNaC::fderivative, 629
  - GiNaC::idx, 725
  - GiNaC::indexed, 744
  - GiNaC::numeric, 939
  - GiNaC::pseries, 1045
  - GiNaC::spinidx, 1103
  - GiNaC::symbol, 1170
  - GiNaC::symmetry, 1181
  - GiNaC::varidx, 1235
  - GiNaC::wildcard, 1242
- do\_renaming
  - GiNaC::make\_flat\_inserter, 796
- domain
  - GiNaC::constant, 453

- dotted
  - GiNaC::spinidx, [1104](#)
- doublefactorial
  - GiNaC, [238](#)
- dummy
  - GiNaC::function\_options, [657](#)
- dump\_hierarchy
  - GiNaC::class\_info< OPT >, [392](#)
- dump\_tree
  - GiNaC::class\_info< OPT >, [392](#)
- duplicate
  - GiNaC::basic, [352](#)
  - GiNaC::possymbol, [982](#)
  - GiNaC::print\_functor\_impl, [1014](#)
  - GiNaC::print\_memfun\_handler< T, C >, [1018](#)
  - GiNaC::print\_ptrfun\_handler< T, C >, [1021](#)
  - GiNaC::realsymbol, [1059](#)
- dynallocate
  - GiNaC, [96](#), [97](#)
- dynallocated
  - GiNaC::status\_flags, [1113](#)
- e
  - GiNaC::archive\_node, [343](#)
  - GiNaC::const\_iterator, [436](#)
  - GiNaC::internal::\_iter\_rep, [306](#)
- Ebar\_kernel
  - GiNaC::Ebar\_kernel, [511](#)
- echelon\_form
  - GiNaC::matrix, [818](#)
- ef
  - GiNaC::constant, [453](#)
- Eisenstein\_h\_kernel
  - GiNaC::Eisenstein\_h\_kernel, [520](#)
- Eisenstein\_kernel
  - GiNaC::Eisenstein\_kernel, [530](#)
- element
  - GiNaC::composition\_generator::coolmulti::element, [534](#)
- ELi\_kernel
  - GiNaC::ELi\_kernel, [541](#)
- EllipticE\_deriv
  - GiNaC, [164](#)
- EllipticE\_eval
  - GiNaC, [164](#)
- EllipticE\_evalf
  - GiNaC, [164](#)
- EllipticE\_print\_latex
  - GiNaC, [164](#)
- EllipticE\_series
  - GiNaC, [164](#)
- EllipticK\_deriv
  - GiNaC, [163](#)
- EllipticK\_eval
  - GiNaC, [163](#)
- EllipticK\_evalf
  - GiNaC, [162](#)
- EllipticK\_print\_latex
  - GiNaC, [163](#)
- EllipticK\_series
  - GiNaC, [163](#)
- empty
  - GiNaC::Gt\_detail::pathintegral\_term\_list, [950](#)
- enable\_tauToFundamental
  - GiNaC::Gt, [709](#)
- end
  - GiNaC::archive\_node::archive\_node\_cit\_range, [345](#)
  - GiNaC::container< C >, [467](#)
  - GiNaC::ex, [556](#)
  - GiNaC::Gt\_detail::pathintegral\_term\_list, [950](#)
- ensure\_if\_modifiable
  - GiNaC::basic, [372](#)
- epp
  - GiNaC, [69](#)
- epsilon\_tensor
  - GiNaC, [274](#), [275](#)
- epvector
  - GiNaC, [69](#)
- equal
  - GiNaC::relational, [1071](#)
- error
  - GiNaC::error\_and\_integral, [545](#)
- error\_and\_integral
  - GiNaC::error\_and\_integral, [545](#)
- eta\_conjugate
  - GiNaC, [152](#)
- eta\_eval
  - GiNaC, [151](#)
- eta\_evalf
  - GiNaC, [151](#)
- eta\_imag\_part
  - GiNaC, [152](#)
- eta\_real\_part
  - GiNaC, [152](#)
- eta\_series
  - GiNaC, [151](#)
- Euler
  - GiNaC, [283](#)
- EulerEvalf
  - GiNaC, [245](#)
- eval
  - GiNaC, [120](#)
  - GiNaC::add, [319](#)
  - GiNaC::basic, [352](#)
  - GiNaC::ex, [557](#)
  - GiNaC::expairseq, [597](#)
  - GiNaC::fderivative, [625](#)
  - GiNaC::function, [642](#)
  - GiNaC::Gt, [700](#)
  - GiNaC::indexed, [739](#)
  - GiNaC::integral, [753](#)
  - GiNaC::mul, [849](#)
  - GiNaC::ncmul, [906](#)
  - GiNaC::numeric, [922](#)
  - GiNaC::power, [989](#)
  - GiNaC::pseries, [1038](#)

- GiNaC::structure< T, ComparisonPolicy >, 1119
  - GiNaC::symbol, 1165
- eval\_f
  - GiNaC::function\_options, 687
- eval\_func
  - GiNaC::function\_options, 658–660, 680
- eval\_funcp
  - GiNaC, 70
- eval\_funcp\_1
  - GiNaC, 71
- eval\_funcp\_10
  - GiNaC, 82
- eval\_funcp\_11
  - GiNaC, 84
- eval\_funcp\_12
  - GiNaC, 85
- eval\_funcp\_13
  - GiNaC, 87
- eval\_funcp\_14
  - GiNaC, 88
- eval\_funcp\_2
  - GiNaC, 72
- eval\_funcp\_3
  - GiNaC, 73
- eval\_funcp\_4
  - GiNaC, 75
- eval\_funcp\_5
  - GiNaC, 76
- eval\_funcp\_6
  - GiNaC, 77
- eval\_funcp\_7
  - GiNaC, 78
- eval\_funcp\_8
  - GiNaC, 80
- eval\_funcp\_9
  - GiNaC, 81
- eval\_funcp\_exvector
  - GiNaC, 90
- eval\_indexed
  - GiNaC::basic, 353
  - GiNaC::matrix, 808
  - GiNaC::minkmetric, 828
  - GiNaC::spinmetric, 1109
  - GiNaC::structure< T, ComparisonPolicy >, 1120
  - GiNaC::su3d, 1137
  - GiNaC::su3f, 1143
  - GiNaC::tensdelta, 1191
  - GiNaC::tensepsilon, 1198
  - GiNaC::tensmetric, 1205
- eval\_integ
  - GiNaC, 121
  - GiNaC::basic, 353
  - GiNaC::ex, 557
  - GiNaC::integral, 756
  - GiNaC::pseries, 1040
- eval\_ncmul
  - GiNaC::add, 322
  - GiNaC::basic, 353
  - GiNaC::clifford, 403
  - GiNaC::color, 422
  - GiNaC::ex, 557
  - GiNaC::function, 643
  - GiNaC::integral, 754
  - GiNaC::mul, 852
  - GiNaC::power, 994
  - GiNaC::relational, 1074
  - GiNaC::structure< T, ComparisonPolicy >, 1119
- eval\_use\_exvector\_args
  - GiNaC::function\_options, 690
- evalchildren
  - GiNaC::expairseq, 606
- evalf
  - GiNaC, 120, 208
  - GiNaC::basic, 352
  - GiNaC::constant, 449
  - GiNaC::ex, 557
  - GiNaC::function, 642
  - GiNaC::Gt, 701
  - GiNaC::idx, 720
  - GiNaC::integral, 753
  - GiNaC::mul, 849
  - GiNaC::numeric, 922
  - GiNaC::power, 990
  - GiNaC::pseries, 1038
  - GiNaC::symbol, 1165
- evalf\_f
  - GiNaC::function\_options, 687
- evalf\_func
  - GiNaC::function\_options, 660–662, 680
- evalf\_funcp
  - GiNaC, 70
- evalf\_funcp\_1
  - GiNaC, 71
- evalf\_funcp\_10
  - GiNaC, 83
- evalf\_funcp\_11
  - GiNaC, 84
- evalf\_funcp\_12
  - GiNaC, 85
- evalf\_funcp\_13
  - GiNaC, 87
- evalf\_funcp\_14
  - GiNaC, 88
- evalf\_funcp\_2
  - GiNaC, 72
- evalf\_funcp\_3
  - GiNaC, 73
- evalf\_funcp\_4
  - GiNaC, 75
- evalf\_funcp\_5
  - GiNaC, 76
- evalf\_funcp\_6
  - GiNaC, 77
- evalf\_funcp\_7
  - GiNaC, 79
- evalf\_funcp\_8

- GiNaC, 80
- evalf\_funcp\_9
  - GiNaC, 81
- evalf\_funcp\_exvector
  - GiNaC, 90
- evalf\_params\_first
  - GiNaC::function\_options, 689
- evalf\_use\_exvector\_args
  - GiNaC::function\_options, 690
- evalffunctype
  - GiNaC, 69
- evalm
  - GiNaC, 121
  - GiNaC::add, 319
  - GiNaC::basic, 352
  - GiNaC::ex, 557
  - GiNaC::matrix, 807
  - GiNaC::mul, 850
  - GiNaC::ncmul, 906
  - GiNaC::power, 990
  - GiNaC::pseries, 1040
  - GiNaC::structure< T, ComparisonPolicy >, 1119
- evaluate
  - GiNaC::Gt, 704
  - GiNaC::Gt\_detail::pathintegral\_term, 948
  - GiNaC::scalar\_products, 1091
- evaluated
  - GiNaC::status\_flags, 1114
- even
  - GiNaC::info\_flags, 747
- ex
  - GiNaC::basic, 374
  - GiNaC::const\_iterator, 435
  - GiNaC::ex, 554, 555
- ex.cpp, 1265
- ex.h, 1265
- ex\_cutIntegrationPath
  - GiNaC::Gt, 705, 706
- ex\_evaluate
  - GiNaC::Gt, 705, 707
- ex\_prepare
  - GiNaC::Gt, 705, 707
- ex\_qExpand
  - GiNaC::Gt, 705, 707
- ex\_regularise
  - GiNaC::Gt, 704, 706
- ex\_tauToFundamental
  - GiNaC::Gt, 705, 706
- ex\_to
  - GiNaC, 125
  - GiNaC::ex, 580
- ex\_zisToFundamental
  - GiNaC::Gt, 704, 706
- exadd
  - GiNaC, 249
- excompiler.cpp, 1268
- excompiler.h, 1268
- exhashmap
  - GiNaC, 91
- exmap
  - GiNaC, 69
- exminus
  - GiNaC, 250
- exmul
  - GiNaC, 249
- exp
  - GiNaC, 230
- exp\_conjugate
  - GiNaC, 178
- exp\_deriv
  - GiNaC, 178
- exp\_eval
  - GiNaC, 177
- exp\_evalf
  - GiNaC, 177
- exp\_expand
  - GiNaC, 178
- exp\_imag\_part
  - GiNaC, 178
- exp\_info
  - GiNaC, 179
- exp\_power
  - GiNaC, 179
- exp\_real\_part
  - GiNaC, 178
- expair
  - GiNaC::expair, 586
- expair.cpp, 1269
- expair.h, 1270
- expair\_needs\_further\_processing
  - GiNaC::expairseq, 602
  - GiNaC::mul, 854
- expairseq
  - GiNaC::expairseq, 595, 596
- expairseq.cpp, 1271
- expairseq.h, 1271
- expand
  - GiNaC, 117, 208
  - GiNaC::add, 324
  - GiNaC::basic, 359
  - GiNaC::ex, 566
  - GiNaC::expairseq, 600
  - GiNaC::function, 642
  - GiNaC::indexed, 742
  - GiNaC::integral, 755
  - GiNaC::mul, 855
  - GiNaC::ncmul, 905
  - GiNaC::power, 995
  - GiNaC::pseries, 1039
  - GiNaC::structure< T, ComparisonPolicy >, 1124
- expand\_add
  - GiNaC::power, 996
- expand\_add\_2
  - GiNaC::power, 997
- expand\_dummy\_sum
  - GiNaC, 140

- expand\_f
  - GiNaC::function\_options, 688
- expand\_func
  - GiNaC::function\_options, 668–670, 681
- expand\_funcp
  - GiNaC, 70
- expand\_funcp\_1
  - GiNaC, 71
- expand\_funcp\_10
  - GiNaC, 83
- expand\_funcp\_11
  - GiNaC, 84
- expand\_funcp\_12
  - GiNaC, 86
- expand\_funcp\_13
  - GiNaC, 87
- expand\_funcp\_14
  - GiNaC, 89
- expand\_funcp\_2
  - GiNaC, 73
- expand\_funcp\_3
  - GiNaC, 74
- expand\_funcp\_4
  - GiNaC, 75
- expand\_funcp\_5
  - GiNaC, 76
- expand\_funcp\_6
  - GiNaC, 78
- expand\_funcp\_7
  - GiNaC, 79
- expand\_funcp\_8
  - GiNaC, 80
- expand\_funcp\_9
  - GiNaC, 82
- expand\_funcp\_exvector
  - GiNaC, 90
- expand\_function\_args
  - GiNaC::expand\_options, 611
- expand\_indexed
  - GiNaC::expand\_options, 610
- expand\_map\_function
  - GiNaC::expand\_map\_function, 609
- expand\_mul
  - GiNaC::power, 997
- expand\_rename\_idx
  - GiNaC::expand\_options, 611
- expand\_transcendental
  - GiNaC::expand\_options, 611
- expand\_use\_exvector\_args
  - GiNaC::function\_options, 691
- expandchildren
  - GiNaC::expairseq, 606
  - GiNaC::mul, 857
  - GiNaC::ncmul, 909
- expanded
  - GiNaC::info\_flags, 747
  - GiNaC::status\_flags, 1114
- expl\_derivative
  - GiNaC::function, 647
- expl\_derivative\_f
  - GiNaC::function\_options, 688
- expl\_derivative\_func
  - GiNaC::function\_options, 672–674, 681
- expl\_derivative\_funcp
  - GiNaC, 70
- expl\_derivative\_funcp\_1
  - GiNaC, 72
- expl\_derivative\_funcp\_10
  - GiNaC, 83
- expl\_derivative\_funcp\_11
  - GiNaC, 85
- expl\_derivative\_funcp\_12
  - GiNaC, 86
- expl\_derivative\_funcp\_13
  - GiNaC, 88
- expl\_derivative\_funcp\_14
  - GiNaC, 89
- expl\_derivative\_funcp\_2
  - GiNaC, 73
- expl\_derivative\_funcp\_3
  - GiNaC, 74
- expl\_derivative\_funcp\_4
  - GiNaC, 75
- expl\_derivative\_funcp\_5
  - GiNaC, 77
- expl\_derivative\_funcp\_6
  - GiNaC, 78
- expl\_derivative\_funcp\_7
  - GiNaC, 79
- expl\_derivative\_funcp\_8
  - GiNaC, 81
- expl\_derivative\_funcp\_9
  - GiNaC, 82
- expl\_derivative\_funcp\_exvector
  - GiNaC, 90
- expl\_derivative\_use\_exvector\_args
  - GiNaC::function\_options, 691
- exponent
  - GiNaC::power, 998
- exponop
  - GiNaC::pseries, 1043
- exprs
  - GiNaC::archive, 333
- exprseq
  - GiNaC, 70
  - GiNaC::info\_flags, 747
- exprseq.cpp, 1272
- exprseq.h, 1273
- exprtable
  - GiNaC::archive, 333
- exset
  - GiNaC, 69
- exvector
  - GiNaC, 69
- exvectorvector
  - GiNaC, 91



## F

- GiNaC::print\_memfun\_handler< T, C >, [1018](#)
  - GiNaC::print\_ptrfun\_handler< T, C >, [1021](#)
- f
  - GiNaC::integral, [759](#)
  - GiNaC::print\_memfun\_handler< T, C >, [1019](#)
  - GiNaC::print\_ptrfun\_handler< T, C >, [1022](#)
  - GiNaC::user\_defined\_kernel, [1226](#)
- factor
  - GiNaC, [130](#)
- factor.cpp, [1273](#)
  - DCOUT, [1274](#)
  - DCOUT2, [1274](#)
  - DCOUTVAR, [1274](#)
  - USE\_SAME\_DEGREE\_FACTOR, [1274](#)
- factor.h, [1275](#)
- factorial
  - GiNaC, [238](#)
- factorial\_conjugate
  - GiNaC, [155](#)
- factorial\_eval
  - GiNaC, [155](#)
- factorial\_evalf
  - GiNaC, [155](#)
- factorial\_imag\_part
  - GiNaC, [156](#)
- factorial\_print\_dflt\_latex
  - GiNaC, [155](#)
- factorial\_real\_part
  - GiNaC, [155](#)
- fail.cpp, [1275](#)
- fail.h, [1276](#)
- FAST\_COMPARE
  - normal.cpp, [1325](#)
- fderivative
  - GiNaC::fderivative, [624](#), [625](#)
  - GiNaC::function\_options, [686](#)
- fderivative.cpp, [1276](#)
- fderivative.h, [1277](#)
- fibonacci
  - GiNaC, [240](#)
- find
  - GiNaC, [118](#)
  - GiNaC::class\_info< OPT >, [392](#)
  - GiNaC::ex, [562](#)
  - GiNaC::unarchive\_table\_t, [1218](#)
- find\_bool
  - GiNaC::archive\_node, [338](#)
- find\_common\_factor
  - GiNaC, [226](#)
- find\_dummy\_indices
  - GiNaC, [134](#)
- find\_ex
  - GiNaC::archive\_node, [340](#)
- find\_ex\_by\_loc
  - GiNaC::archive\_node, [340](#)
- find\_ex\_node
  - GiNaC::archive\_node, [340](#)
- find\_factory\_fcn
  - GiNaC, [94](#)
- find\_first
  - GiNaC::archive\_node, [339](#)
- find\_free\_and\_dummy
  - GiNaC, [133](#), [134](#)
- find\_function
  - GiNaC::function, [648](#)
- find\_last
  - GiNaC::archive\_node, [339](#)
- find\_property\_range
  - GiNaC::archive\_node, [339](#)
- find\_real\_imag
  - GiNaC::mul, [856](#)
- find\_string
  - GiNaC::archive\_node, [339](#)
- find\_unsigned
  - GiNaC::archive\_node, [338](#)
- find\_variant\_indices
  - GiNaC, [137](#)
- findMoebiusTransform
  - GiNaC::Gt, [703](#)
- finished
  - GiNaC::composition\_generator::coolmulti, [474](#)
- first
  - GiNaC::class\_info< OPT >, [393](#)
- flag\_overflow
  - GiNaC::basic\_multi\_iterator< T >, [388](#)
- flags
  - GiNaC::basic, [375](#)
- flags.h, [1277](#)
- force\_include\_tgamma
  - GiNaC, [284](#)
- force\_include\_zeta1
  - GiNaC, [284](#)
- forget
  - GiNaC::archive, [331](#)
  - GiNaC::archive\_node, [342](#)
- format\_index\_value
  - GiNaC, [278](#), [279](#)
- frac\_cancel
  - GiNaC, [226](#)
- fraction\_free\_elimination
  - GiNaC::matrix, [819](#)
- fsolve
  - GiNaC, [159](#)
- func\_arg\_info
  - GiNaC, [142](#)
- func\_default
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, [1216](#)
- func\_obj
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, [1216](#)
- FUNCP\_1P
  - GiNaC, [69](#)
- FUNCP\_2P
  - GiNaC, [69](#)

- FUNCP\_CUBA
  - GiNaC, [69](#)
- function
  - GiNaC::function, [637–641](#)
  - GiNaC::function\_options, [686](#)
- function.cpp, [1278](#)
- function.h, [1279](#)
  - DECLARE\_FUNCTION\_10P, [1288](#)
  - DECLARE\_FUNCTION\_11P, [1288](#)
  - DECLARE\_FUNCTION\_12P, [1288](#)
  - DECLARE\_FUNCTION\_13P, [1289](#)
  - DECLARE\_FUNCTION\_14P, [1289](#)
  - DECLARE\_FUNCTION\_1P, [1286](#)
  - DECLARE\_FUNCTION\_2P, [1286](#)
  - DECLARE\_FUNCTION\_3P, [1286](#)
  - DECLARE\_FUNCTION\_4P, [1286](#)
  - DECLARE\_FUNCTION\_5P, [1287](#)
  - DECLARE\_FUNCTION\_6P, [1287](#)
  - DECLARE\_FUNCTION\_7P, [1287](#)
  - DECLARE\_FUNCTION\_8P, [1287](#)
  - DECLARE\_FUNCTION\_9P, [1288](#)
  - is\_ex\_the\_function, [1289](#)
  - REGISTER\_FUNCTION, [1289](#)
- function\_options
  - GiNaC::function\_options, [656, 657](#)
- functions\_with\_same\_name
  - GiNaC::function\_options, [692](#)
- G
  - GiNaC, [160](#)
- G2\_eval
  - GiNaC, [171](#)
- G2\_evalf
  - GiNaC, [171](#)
- G3\_eval
  - GiNaC, [172](#)
- G3\_evalf
  - GiNaC, [172](#)
- G\_path
  - GiNaC::Gt\_detail::pathintegral\_term, [948](#)
- gauss
  - GiNaC::determinant\_algo, [477](#)
  - GiNaC::solve\_algo, [1093](#)
- gauss\_elimination
  - GiNaC::matrix, [818](#)
- gcd
  - GiNaC, [222, 243](#)
- gcd\_pf\_mul
  - GiNaC, [221](#)
- gcd\_pf\_pow
  - GiNaC, [221](#)
- gcd\_pf\_pow\_pow
  - GiNaC, [222](#)
- generalised\_Bernoulli\_number
  - GiNaC, [201](#)
- get
  - GiNaC::composition\_generator, [429](#)
  - GiNaC::partition\_generator, [942](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [945](#)
- get\_all\_dummy\_indices
  - GiNaC, [139](#)
- get\_all\_dummy\_indices\_safely
  - GiNaC, [139](#)
- get\_cache
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, [1215](#)
- get\_cache\_size
  - GiNaC::integration\_kernel, [766](#)
- get\_class\_name
  - GiNaC::structure< T, ComparisonPolicy >, [1119](#)
- get\_clifford\_comp
  - GiNaC, [107](#)
- get\_close\_delim
  - GiNaC::container< C >, [461](#)
- get\_commutator\_sign
  - GiNaC::clifford, [405](#)
- get\_default\_flags
  - GiNaC::container< C >, [461](#)
- get\_default\_TeX\_name
  - GiNaC, [263](#)
- get\_dim
  - GiNaC::idx, [724](#)
- get\_dim\_uint
  - GiNaC, [99](#)
- get\_domain
  - GiNaC::possymbol, [982](#)
  - GiNaC::realsymbol, [1059](#)
  - GiNaC::symbol, [1169](#)
- get\_dummy\_indices
  - GiNaC::indexed, [742, 743](#)
- get\_ex
  - GiNaC::archive\_node, [341](#)
- get\_factors
  - GiNaC::ncmul, [910](#)
- get\_first\_symbol
  - GiNaC, [212](#)
- get\_free\_indices
  - GiNaC::add, [322](#)
  - GiNaC::basic, [363](#)
  - GiNaC::ex, [573](#)
  - GiNaC::indexed, [740](#)
  - GiNaC::integral, [755](#)
  - GiNaC::mul, [852](#)
  - GiNaC::ncmul, [907](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1128](#)
- get\_id
  - GiNaC::print\_context\_options, [1002](#)
  - GiNaC::registered\_class\_options, [1064](#)
- get\_indices
  - GiNaC::indexed, [742](#)
- get\_label
  - GiNaC::wildcard, [1241](#)
- get\_last\_access
  - GiNaC::remember\_table\_entry, [1083](#)
- get\_metric
  - GiNaC::clifford, [405](#)
- get\_name

- GiNaC::constant, [451](#)
  - GiNaC::function, [649](#)
  - GiNaC::function\_options, [685](#)
  - GiNaC::print\_context\_options, [1001](#)
  - GiNaC::registered\_class\_options, [1064](#)
  - GiNaC::symbol, [1170](#)
- get\_node
  - GiNaC::archive, [331](#)
- get\_nparams
  - GiNaC::function\_options, [685](#)
- get\_numerical\_value
  - GiNaC::Ebar\_kernel, [512](#)
  - GiNaC::Eisenstein\_h\_kernel, [521](#)
  - GiNaC::Eisenstein\_kernel, [531](#)
  - GiNaC::ELi\_kernel, [542](#)
  - GiNaC::integration\_kernel, [765](#)
  - GiNaC::Kronecker\_dtau\_kernel, [780](#)
  - GiNaC::Kronecker\_dz\_kernel, [789](#)
  - GiNaC::modular\_form\_kernel, [837](#)
- get\_numerical\_value\_impl
  - GiNaC::integration\_kernel, [767](#)
- get\_open\_delim
  - GiNaC::container< C >, [461](#)
- get\_order
  - GiNaC::lanczos\_coeffs, [791](#)
- get\_parent
  - GiNaC::class\_info< OPT >, [392](#)
- get\_parent\_name
  - GiNaC::print\_context\_options, [1001](#)
  - GiNaC::registered\_class\_options, [1064](#)
- get\_point
  - GiNaC::pseries, [1041](#)
- get\_pointer
  - GiNaC::ptr< T >, [1052](#)
- get\_print\_context
  - GiNaC, [256](#)
- get\_print\_dispatch\_table
  - GiNaC::registered\_class\_options, [1064](#)
- get\_print\_options
  - GiNaC, [257](#)
- get\_properties
  - GiNaC::archive\_node, [341](#)
- get\_refcount
  - GiNaC::refcounted, [1062](#)
- get\_registered\_functions
  - GiNaC::function, [648](#)
- get\_representation\_label
  - GiNaC, [102](#), [114](#)
  - GiNaC::clifford, [405](#)
  - GiNaC::color, [424](#)
- get\_result
  - GiNaC::remember\_table\_entry, [1083](#)
- get\_serial
  - GiNaC::function, [649](#)
- get\_series\_coeff
  - GiNaC::integration\_kernel, [766](#)
- get\_sign
  - GiNaC::multi\_iterator\_permutation< T >, [881](#)
- get\_struct
  - GiNaC::structure< T, ComparisonPolicy >, [1132](#)
- get\_successful\_hits
  - GiNaC::remember\_table\_entry, [1083](#)
- get\_symbol\_stats
  - GiNaC, [214](#)
- get\_symmetry
  - GiNaC::indexed, [743](#)
- get\_TeX\_name
  - GiNaC::constant, [451](#)
  - GiNaC::symbol, [1170](#)
- get\_top\_node
  - GiNaC::archive, [331](#)
- get\_type
  - GiNaC::symmetry, [1180](#)
- get\_value
  - GiNaC::idx, [723](#)
- get\_var
  - GiNaC::pseries, [1041](#)
- get\_vector
  - GiNaC::basic\_multi\_iterator< T >, [385](#)
- getArgs
  - GiNaC::Gt, [700](#)
- gethash
  - GiNaC::basic, [371](#)
  - GiNaC::ex, [577](#)
- getTau
  - GiNaC::Gt, [700](#)
- getZ
  - GiNaC::Gt, [700](#)
- GiNaC, [29](#)
  - \_ex0, [292](#)
  - \_ex1, [294](#)
  - \_ex10, [297](#)
  - \_ex11, [297](#)
  - \_ex12, [298](#)
  - \_ex120, [300](#)
  - \_ex15, [298](#)
  - \_ex18, [298](#)
  - \_ex1\_2, [294](#)
  - \_ex1\_3, [293](#)
  - \_ex1\_4, [293](#)
  - \_ex2, [295](#)
  - \_ex20, [299](#)
  - \_ex24, [299](#)
  - \_ex25, [299](#)
  - \_ex3, [295](#)
  - \_ex30, [299](#)
  - \_ex4, [295](#)
  - \_ex48, [300](#)
  - \_ex5, [296](#)
  - \_ex6, [296](#)
  - \_ex60, [300](#)
  - \_ex7, [296](#)
  - \_ex8, [297](#)
  - \_ex9, [297](#)
  - \_ex\_1, [291](#)
  - \_ex\_10, [289](#)

\_ex\_11, 289  
 \_ex\_12, 288  
 \_ex\_120, 286  
 \_ex\_15, 288  
 \_ex\_18, 288  
 \_ex\_1\_2, 292  
 \_ex\_1\_3, 292  
 \_ex\_1\_4, 292  
 \_ex\_2, 291  
 \_ex\_20, 288  
 \_ex\_24, 287  
 \_ex\_25, 287  
 \_ex\_3, 291  
 \_ex\_30, 287  
 \_ex\_4, 290  
 \_ex\_48, 287  
 \_ex\_5, 290  
 \_ex\_6, 290  
 \_ex\_60, 286  
 \_ex\_7, 290  
 \_ex\_8, 289  
 \_ex\_9, 289  
 \_num0\_bp, 284  
 \_num0\_p, 292  
 \_num10\_p, 297  
 \_num11\_p, 297  
 \_num120\_p, 300  
 \_num12\_p, 298  
 \_num15\_p, 298  
 \_num18\_p, 298  
 \_num1\_2\_p, 293  
 \_num1\_3\_p, 293  
 \_num1\_4\_p, 293  
 \_num1\_p, 294  
 \_num20\_p, 298  
 \_num24\_p, 299  
 \_num25\_p, 299  
 \_num2\_p, 295  
 \_num30\_p, 299  
 \_num3\_p, 295  
 \_num48\_p, 299  
 \_num4\_p, 295  
 \_num5\_p, 296  
 \_num60\_p, 300  
 \_num6\_p, 296  
 \_num7\_p, 296  
 \_num8\_p, 296  
 \_num9\_p, 297  
 \_num\_10\_p, 289  
 \_num\_11\_p, 288  
 \_num\_120\_p, 286  
 \_num\_12\_p, 288  
 \_num\_15\_p, 288  
 \_num\_18\_p, 288  
 \_num\_1\_2\_p, 291  
 \_num\_1\_3\_p, 292  
 \_num\_1\_4\_p, 292  
 \_num\_1\_p, 291  
 \_num\_20\_p, 287  
 \_num\_24\_p, 287  
 \_num\_25\_p, 287  
 \_num\_2\_p, 291  
 \_num\_30\_p, 287  
 \_num\_3\_p, 290  
 \_num\_48\_p, 286  
 \_num\_4\_p, 290  
 \_num\_5\_p, 290  
 \_num\_60\_p, 286  
 \_num\_6\_p, 290  
 \_num\_7\_p, 289  
 \_num\_8\_p, 289  
 \_num\_9\_p, 289  
 abs, 240  
 abs\_conjugate, 147  
 abs\_eval, 146  
 abs\_evalf, 146  
 abs\_expand, 146  
 abs\_expl\_derivative, 146  
 abs\_imag\_part, 147  
 abs\_info, 148  
 abs\_power, 147  
 abs\_print\_csrc\_float, 147  
 abs\_print\_latex, 146  
 abs\_real\_part, 147  
 acos, 232  
 acos\_conjugate, 188  
 acos\_deriv, 187  
 acos\_eval, 187  
 acos\_evalf, 187  
 acosh, 235  
 acosh\_conjugate, 196  
 acosh\_deriv, 196  
 acosh\_eval, 196  
 acosh\_evalf, 196  
 adaptivesimpson, 198  
 add\_symbol, 214  
 algebraic\_match\_mul\_with\_mul, 211  
 antisymmetric2, 266  
 antisymmetric3, 266  
 antisymmetric4, 266  
 antisymmetrize, 122, 123, 267, 271  
 archive\_atom, 68  
 archive\_node\_id, 68  
 are\_ex\_trivially\_equal, 116  
 asin, 232  
 asin\_conjugate, 186  
 asin\_deriv, 186  
 asin\_eval, 186  
 asin\_evalf, 186  
 asin\_info, 187  
 asinh, 234  
 asinh\_conjugate, 195  
 asinh\_deriv, 195  
 asinh\_eval, 195  
 asinh\_evalf, 195  
 atan, 232, 233

atan2\_deriv, 190  
atan2\_eval, 190  
atan2\_evalf, 189  
atan2\_info, 190  
atan\_conjugate, 189  
atan\_deriv, 188  
atan\_eval, 188  
atan\_evalf, 188  
atan\_info, 189  
atan\_series, 189  
atanh, 235  
atanh\_conjugate, 198  
atanh\_deriv, 197  
atanh\_eval, 197  
atanh\_evalf, 197  
atanh\_series, 197  
base\_and\_index, 98  
bernoulli, 239  
Bernoulli\_polynomial, 201  
beta\_deriv, 169  
beta\_eval, 169  
beta\_evalf, 169  
beta\_series, 169  
binomial, 239  
binomial\_conjugate, 157  
binomial\_eval, 156  
binomial\_evalf, 156  
binomial\_imag\_part, 157  
binomial\_real\_part, 157  
binomial\_sym, 156  
callback\_registered, 92  
canonicalize, 267  
canonicalize\_clifford, 103  
Catalan, 283  
CatalanEvalf, 245  
charpoly, 209  
class\_info< OPT >::first, 282  
class\_info< OPT >::parents\_identified, 283  
clifford\_bar, 110  
clifford\_inverse, 105  
clifford\_max\_label, 104  
clifford\_moebius\_map, 107, 108  
clifford\_norm, 105  
clifford\_prime, 104  
clifford\_star, 110  
clifford\_star\_bar, 103  
clifford\_to\_lst, 107  
clifford\_unit, 99  
coeff, 119  
coerce, 229  
coerce< int, cln::cl\_I >, 229  
coerce< unsigned int, cln::cl\_I >, 229  
collect, 120  
collect\_common\_factors, 226  
collect\_symbols, 214  
color\_d, 113  
color\_f, 112  
color\_h, 113  
color\_ONE, 111  
color\_T, 112  
color\_trace, 114, 115  
cols, 209  
compare\_pointers, 277  
compile\_ex, 127, 128  
conjugate, 117  
conjugate\_conjugate, 141  
conjugate\_eval, 141  
conjugate\_evalf, 141  
conjugate\_expl\_derivative, 141  
conjugate\_funcp, 70  
conjugate\_funcp\_1, 71  
conjugate\_funcp\_10, 83  
conjugate\_funcp\_11, 84  
conjugate\_funcp\_12, 85  
conjugate\_funcp\_13, 87  
conjugate\_funcp\_14, 88  
conjugate\_funcp\_2, 72  
conjugate\_funcp\_3, 74  
conjugate\_funcp\_4, 75  
conjugate\_funcp\_5, 76  
conjugate\_funcp\_6, 77  
conjugate\_funcp\_7, 79  
conjugate\_funcp\_8, 80  
conjugate\_funcp\_9, 81  
conjugate\_funcp\_exvector, 90  
conjugate\_imag\_part, 142  
conjugate\_info, 142  
conjugate\_print\_latex, 141  
conjugate\_real\_part, 142  
conjugateepvector, 129  
convert\_H\_to\_Li, 162  
cos, 231  
cos\_conjugate, 184  
cos\_deriv, 183  
cos\_eval, 183  
cos\_evalf, 183  
cos\_imag\_part, 184  
cos\_real\_part, 183  
cosh, 234  
cosh\_conjugate, 193  
cosh\_deriv, 192  
cosh\_eval, 192  
cosh\_evalf, 192  
cosh\_imag\_part, 193  
cosh\_real\_part, 192  
count\_dummy\_indices, 136  
count\_free\_indices, 136  
crc32, 116  
crctab, 284  
csgn, 246  
csgn\_conjugate, 150  
csgn\_eval, 150  
csgn\_evalf, 149  
csgn\_imag\_part, 150  
csgn\_power, 151  
csgn\_real\_part, 150

csgn\_series, 150  
csrc, 258  
csrc\_cl\_N, 259  
csrc\_double, 259  
csrc\_float, 258  
cyclic\_permutation, 278  
decomp\_rational, 216  
degree, 118  
delta\_tensor, 272  
denom, 119, 249  
derivative\_funcp, 70  
derivative\_funcp\_1, 72  
derivative\_funcp\_10, 83  
derivative\_funcp\_11, 84  
derivative\_funcp\_12, 86  
derivative\_funcp\_13, 87  
derivative\_funcp\_14, 89  
derivative\_funcp\_2, 73  
derivative\_funcp\_3, 74  
derivative\_funcp\_4, 75  
derivative\_funcp\_5, 76  
derivative\_funcp\_6, 78  
derivative\_funcp\_7, 79  
derivative\_funcp\_8, 80  
derivative\_funcp\_9, 82  
derivative\_funcp\_exvector, 90  
determinant, 209  
dfft, 258  
diag\_matrix, 206, 207  
diff, 121  
Digits, 285  
digits\_changed\_callback, 92  
dirac\_gamma, 100  
dirac\_gamma5, 100  
dirac\_gammaL, 100  
dirac\_gammaR, 101  
dirac\_ONE, 99  
dirac\_slash, 101  
dirac\_trace, 102, 103  
dirichlet\_character, 200  
divide, 218  
divide\_in\_z, 218  
doublefactorial, 238  
dynallocate, 96, 97  
EllipticE\_deriv, 164  
EllipticE\_eval, 164  
EllipticE\_evalf, 164  
EllipticE\_print\_latex, 164  
EllipticE\_series, 164  
EllipticK\_deriv, 163  
EllipticK\_eval, 163  
EllipticK\_evalf, 162  
EllipticK\_print\_latex, 163  
EllipticK\_series, 163  
epp, 69  
epsilon\_tensor, 274, 275  
epvector, 69  
eta\_conjugate, 152  
eta\_eval, 151  
eta\_evalf, 151  
eta\_imag\_part, 152  
eta\_real\_part, 152  
eta\_series, 151  
Euler, 283  
EulerEvalf, 245  
eval, 120  
eval\_funcp, 70  
eval\_funcp\_1, 71  
eval\_funcp\_10, 82  
eval\_funcp\_11, 84  
eval\_funcp\_12, 85  
eval\_funcp\_13, 87  
eval\_funcp\_14, 88  
eval\_funcp\_2, 72  
eval\_funcp\_3, 73  
eval\_funcp\_4, 75  
eval\_funcp\_5, 76  
eval\_funcp\_6, 77  
eval\_funcp\_7, 78  
eval\_funcp\_8, 80  
eval\_funcp\_9, 81  
eval\_funcp\_exvector, 90  
eval\_integ, 121  
evalf, 120, 208  
evalf\_funcp, 70  
evalf\_funcp\_1, 71  
evalf\_funcp\_10, 83  
evalf\_funcp\_11, 84  
evalf\_funcp\_12, 85  
evalf\_funcp\_13, 87  
evalf\_funcp\_14, 88  
evalf\_funcp\_2, 72  
evalf\_funcp\_3, 73  
evalf\_funcp\_4, 75  
evalf\_funcp\_5, 76  
evalf\_funcp\_6, 77  
evalf\_funcp\_7, 79  
evalf\_funcp\_8, 80  
evalf\_funcp\_9, 81  
evalf\_funcp\_exvector, 90  
evalffuncdtype, 69  
evalm, 121  
ex\_to, 125  
exadd, 249  
exhashmap, 91  
exmap, 69  
exminus, 250  
exmul, 249  
exp, 230  
exp\_conjugate, 178  
exp\_deriv, 178  
exp\_eval, 177  
exp\_evalf, 177  
exp\_expand, 178  
exp\_imag\_part, 178  
exp\_info, 179

- exp\_power, 179
- exp\_real\_part, 178
- expand, 117, 208
- expand\_dummy\_sum, 140
- expand\_funcp, 70
- expand\_funcp\_1, 71
- expand\_funcp\_10, 83
- expand\_funcp\_11, 84
- expand\_funcp\_12, 86
- expand\_funcp\_13, 87
- expand\_funcp\_14, 89
- expand\_funcp\_2, 73
- expand\_funcp\_3, 74
- expand\_funcp\_4, 75
- expand\_funcp\_5, 76
- expand\_funcp\_6, 78
- expand\_funcp\_7, 79
- expand\_funcp\_8, 80
- expand\_funcp\_9, 82
- expand\_funcp\_exvector, 90
- expl\_derivative\_funcp, 70
- expl\_derivative\_funcp\_1, 72
- expl\_derivative\_funcp\_10, 83
- expl\_derivative\_funcp\_11, 85
- expl\_derivative\_funcp\_12, 86
- expl\_derivative\_funcp\_13, 88
- expl\_derivative\_funcp\_14, 89
- expl\_derivative\_funcp\_2, 73
- expl\_derivative\_funcp\_3, 74
- expl\_derivative\_funcp\_4, 75
- expl\_derivative\_funcp\_5, 77
- expl\_derivative\_funcp\_6, 78
- expl\_derivative\_funcp\_7, 79
- expl\_derivative\_funcp\_8, 81
- expl\_derivative\_funcp\_9, 82
- expl\_derivative\_funcp\_exvector, 90
- exprseq, 70
- exset, 69
- exvector, 69
- exvectorvector, 91
- factor, 130
- factorial, 238
- factorial\_conjugate, 155
- factorial\_eval, 155
- factorial\_evalf, 155
- factorial\_imag\_part, 156
- factorial\_print\_dflt\_latex, 155
- factorial\_real\_part, 155
- fibonacci, 240
- find, 118
- find\_common\_factor, 226
- find\_dummy\_indices, 134
- find\_factory\_fcn, 94
- find\_free\_and\_dummy, 133, 134
- find\_variant\_indices, 137
- force\_include\_tgamma, 284
- force\_include\_zeta1, 284
- format\_index\_value, 278, 279
- frac\_cancel, 226
- fsolve, 159
- func\_arg\_info, 142
- FUNCP\_1P, 69
- FUNCP\_2P, 69
- FUNCP\_CUBA, 69
- G, 160
- G2\_eval, 171
- G2\_evalf, 171
- G3\_eval, 172
- G3\_evalf, 172
- gcd, 222, 243
- gcd\_pf\_mul, 221
- gcd\_pf\_pow, 221
- gcd\_pf\_pow\_pow, 222
- generalised\_Bernoulli\_number, 201
- get\_all\_dummy\_indices, 139
- get\_all\_dummy\_indices\_safely, 139
- get\_clifford\_comp, 107
- get\_default\_TeX\_name, 263
- get\_dim\_uint, 99
- get\_first\_symbol, 212
- get\_print\_context, 256
- get\_print\_options, 257
- get\_representation\_label, 102, 114
- get\_symbol\_stats, 214
- GINAC\_BIND\_UNARCHIVER, 92, 97, 98, 110, 111, 116, 131, 132, 136, 199, 201–204, 206, 211, 212, 228, 260, 261, 263–265, 272, 281, 284
- GINAC\_DECLARE\_UNARCHIVER, 93, 109, 115, 116, 130–132, 134, 141, 199, 204–206, 208, 212, 245, 260, 261, 263, 264, 268, 276, 282
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, 92, 94, 97, 110, 115, 129–132, 136, 198, 201–204, 206, 211, 212, 227, 259, 261, 263, 264, 271, 281
- golden\_ratio\_hash, 277
- guess\_precision, 237
- H\_deriv, 175
- H\_eval, 175
- H\_evalf, 175
- H\_print\_latex, 175
- H\_series, 175
- has, 118
- hasindex, 138
- haswild, 282
- heur\_gcd, 220
- heur\_gcd\_z, 220
- hold\_ncmul, 212
- I, 285
- idx, 284
- idx\_symmetrization, 138
- ifactor, 199
- imag, 249
- imag\_part, 118
- imag\_part\_conjugate, 145
- imag\_part\_eval, 144

- imag\_part\_evalf, 144
- imag\_part\_expl\_derivative, 145
- imag\_part\_funcp, 70
- imag\_part\_funcp\_1, 71
- imag\_part\_funcp\_10, 83
- imag\_part\_funcp\_11, 84
- imag\_part\_funcp\_12, 86
- imag\_part\_funcp\_13, 87
- imag\_part\_funcp\_14, 89
- imag\_part\_funcp\_2, 73
- imag\_part\_funcp\_3, 74
- imag\_part\_funcp\_4, 75
- imag\_part\_funcp\_5, 76
- imag\_part\_funcp\_6, 78
- imag\_part\_funcp\_7, 79
- imag\_part\_funcp\_8, 80
- imag\_part\_funcp\_9, 82
- imag\_part\_funcp\_exvector, 90
- imag\_part\_imag\_part, 145
- imag\_part\_print\_latex, 145
- imag\_part\_real\_part, 145
- index0, 265
- index1, 265
- index2, 265
- index3, 265
- index\_dimensions, 259
- indices\_consistent, 136
- info\_funcp, 71
- info\_funcp\_1, 72
- info\_funcp\_10, 84
- info\_funcp\_11, 85
- info\_funcp\_12, 87
- info\_funcp\_13, 88
- info\_funcp\_14, 90
- info\_funcp\_2, 73
- info\_funcp\_3, 74
- info\_funcp\_4, 76
- info\_funcp\_5, 77
- info\_funcp\_6, 78
- info\_funcp\_7, 80
- info\_funcp\_8, 81
- info\_funcp\_9, 82
- info\_funcp\_exvector, 91
- interpolate, 219
- inverse, 210, 245
- iquo, 242, 243
- irem, 241, 242
- is\_a, 94, 125, 261
- is\_cinteger, 248
- is\_clifford\_tinfo, 109
- is\_color\_tinfo, 113
- is\_crational, 248
- is\_dirac\_slash, 98
- is\_discriminant\_of\_quadratic\_number\_field, 199
- is\_dummy\_pair, 133
- is\_even, 247
- is\_exactly\_a, 95, 125
- is\_integer, 246
- is\_negative, 246
- is\_nonneg\_integer, 247
- is\_odd, 247
- is\_order\_function, 162
- is\_polynomial, 118
- is\_pos\_integer, 247
- is\_positive, 246
- is\_prime, 247
- is\_rational, 247
- is\_real, 248
- is\_terminating, 262
- is\_the\_function, 131
- is\_the\_function< G\_SERIAL >, 161
- is\_the\_function< iterated\_integral\_SERIAL >, 162
- is\_the\_function< psi\_SERIAL >, 161
- is\_the\_function< zeta\_SERIAL >, 160
- is\_zero, 124, 246
- isqrt, 244
- iterated\_integral, 161, 162
- iterated\_integral2\_eval, 165
- iterated\_integral2\_evalf, 165
- iterated\_integral3\_eval, 166
- iterated\_integral3\_evalf, 165
- iterated\_integral\_evalf\_impl, 165
- kronecker\_symbol, 200
- latex, 258
- lcm, 223, 243
- lcm\_of\_coefficients\_denominators, 215
- lcmcoeff, 214
- ldegree, 119
- lgamma, 237
- lgamma\_conjugate, 167
- lgamma\_deriv, 166
- lgamma\_eval, 166
- lgamma\_evalf, 166
- lgamma\_series, 167
- lhs, 123
- Li2, 236
- Li2\_, 236
- Li2\_conjugate, 153
- Li2\_deriv, 153
- Li2\_eval, 153
- Li2\_evalf, 152
- Li2\_projection, 236
- Li2\_series, 153, 235
- Li3\_eval, 154
- Li\_deriv, 173
- Li\_eval, 172
- Li\_evalf, 172
- Li\_print\_latex, 173
- Li\_series, 172
- library\_initializer, 284
- link\_ex, 128, 129
- log, 230
- log2, 276
- log\_conjugate, 181
- log\_deriv, 180
- log\_eval, 179



log\_evalf, 179  
log\_expand, 180  
log\_imag\_part, 180  
log\_info, 181  
log\_real\_part, 180  
log\_series, 180  
lookup\_map, 91  
lorentz\_eps, 275  
lorentz\_g, 273  
lsolve, 159  
lst, 91  
lst\_to\_clifford, 105  
lst\_to\_matrix, 206  
make\_real\_float, 227  
make\_return\_type\_t, 262  
map\_eval\_integ, 282  
map\_evalm, 282  
match, 121  
metric\_tensor, 273  
minimal\_dim, 133  
mod, 241  
multinomial\_coefficient, 276  
multiply\_lcm, 215  
my\_ios\_callback, 256  
my\_ios\_index, 256  
next\_print\_context\_id, 285  
no\_index\_dimensions, 259  
nops, 117, 208  
normal, 120  
not\_symmetric, 265  
number\_of\_type, 136  
numer, 119, 249  
numer\_denom, 119  
op, 123  
operator!=, 255  
operator<, 255  
operator<<, 93, 116, 117, 245, 257, 279–281  
operator<=, 256  
operator>, 256  
operator>>, 93, 94, 257  
operator>=, 256  
operator+, 250, 251, 253  
operator++, 253–255  
operator+=", 251, 252  
operator-, 250, 251, 253  
operator--, 253–255  
operator=, 251, 252  
operator/, 250, 251  
operator/=", 252  
operator==, 255  
operator\*, 250, 251  
operator\*=", 252  
Order\_conjugate, 158  
Order\_eval, 157  
Order\_expl\_derivative, 158  
Order\_imag\_part, 158  
Order\_power, 158  
Order\_real\_part, 158  
Order\_series, 158  
paramset, 70  
permutation\_sign, 277, 278  
permute\_free\_index\_to\_front, 111  
Pi, 283  
PiEvalf, 244  
pow, 245, 260  
power\_funcp, 71  
power\_funcp\_1, 72  
power\_funcp\_10, 83  
power\_funcp\_11, 85  
power\_funcp\_12, 86  
power\_funcp\_13, 88  
power\_funcp\_14, 89  
power\_funcp\_2, 73  
power\_funcp\_3, 74  
power\_funcp\_4, 75  
power\_funcp\_5, 77  
power\_funcp\_6, 78  
power\_funcp\_7, 79  
power\_funcp\_8, 81  
power\_funcp\_9, 82  
power\_funcp\_exvector, 90  
prem, 217  
primitive\_dirichlet\_character, 200  
print\_context\_class\_info, 92  
print\_func< print\_context >, 132  
print\_func< print\_dflt >, 97, 110, 271  
print\_funcp, 71  
print\_funcp\_1, 72  
print\_funcp\_10, 84  
print\_funcp\_11, 85  
print\_funcp\_12, 86  
print\_funcp\_13, 88  
print\_funcp\_14, 89  
print\_funcp\_2, 73  
print\_funcp\_3, 74  
print\_funcp\_4, 76  
print\_funcp\_5, 77  
print\_funcp\_6, 78  
print\_funcp\_7, 80  
print\_funcp\_8, 81  
print\_funcp\_9, 82  
print\_funcp\_exvector, 91  
print\_integer\_csrc, 228  
print\_operator, 263  
print\_real\_cl\_N, 230  
print\_real\_csrc, 229  
print\_real\_number, 228  
print\_sym\_pow, 259  
product\_to\_exvector, 138  
psi, 161, 238  
psi1\_deriv, 170  
psi1\_eval, 170  
psi1\_evalf, 170  
psi1\_series, 170  
psi2\_deriv, 171  
psi2\_eval, 171

- psi2\_evalf, 170
- psi2\_series, 171
- python, 258
- python\_repr, 258
- quo, 215
- rank, 210
- read\_real\_float, 227
- read\_unsigned, 93
- real, 249
- real\_part, 117
- real\_part\_conjugate, 143
- real\_part\_eval, 143
- real\_part\_evalf, 143
- real\_part\_expl\_derivative, 144
- real\_part\_funcp, 70
- real\_part\_funcp\_1, 71
- real\_part\_funcp\_10, 83
- real\_part\_funcp\_11, 84
- real\_part\_funcp\_12, 86
- real\_part\_funcp\_13, 87
- real\_part\_funcp\_14, 89
- real\_part\_funcp\_2, 72
- real\_part\_funcp\_3, 74
- real\_part\_funcp\_4, 75
- real\_part\_funcp\_5, 76
- real\_part\_funcp\_6, 77
- real\_part\_funcp\_7, 79
- real\_part\_funcp\_8, 80
- real\_part\_funcp\_9, 81
- real\_part\_funcp\_exvector, 90
- real\_part\_imag\_part, 144
- real\_part\_print\_latex, 143
- real\_part\_real\_part, 143
- reduced\_matrix, 207
- reeval\_ncmul, 212
- REGISTER\_FUNCTION, 142, 144, 145, 148, 149, 151–154, 156, 157, 159, 163, 165, 167–169, 173, 174, 176, 179, 181, 183, 184, 186–190, 192, 193, 195–198
- registered\_class\_info, 92
- rem, 216
- remove\_dirac\_ONE, 104
- rename\_dummy\_indices, 137
- rename\_dummy\_indices\_uniquely, 139, 140
- replace\_with\_symbol, 225
- reposition\_dummy\_indices, 137
- resultant, 227
- rhs, 123
- rotate\_left, 277
- rows, 209
- S\_deriv, 174
- S\_eval, 173
- S\_evalf, 173
- S\_print\_latex, 174
- S\_series, 174
- series, 121
- series\_funcp, 71
- series\_funcp\_1, 72
- series\_funcp\_10, 83
- series\_funcp\_11, 85
- series\_funcp\_12, 86
- series\_funcp\_13, 88
- series\_funcp\_14, 89
- series\_funcp\_2, 73
- series\_funcp\_3, 74
- series\_funcp\_4, 75
- series\_funcp\_5, 77
- series\_funcp\_6, 78
- series\_funcp\_7, 79
- series\_funcp\_8, 81
- series\_funcp\_9, 82
- series\_funcp\_exvector, 91
- series\_to\_poly, 261
- set\_print\_context, 257
- set\_print\_func, 262
- set\_print\_options, 257
- shaker\_sort, 278
- simplify\_indexed, 122, 138
- simplify\_indexed\_product, 138
- sin, 231
- sin\_conjugate, 182
- sin\_deriv, 182
- sin\_eval, 181
- sin\_evalf, 181
- sin\_imag\_part, 182
- sin\_real\_part, 182
- sinh, 233
- sinh\_conjugate, 191
- sinh\_deriv, 191
- sinh\_eval, 191
- sinh\_evalf, 190
- sinh\_imag\_part, 191
- sinh\_real\_part, 191
- smod, 241
- spinor\_metric, 274
- spmap, 91
- sprem, 217
- sqrfree, 223
- sqrfree\_parfrac, 224
- sqrfree\_yun, 223
- sqrt, 244, 260
- sr\_gcd, 219
- step, 246
- step\_conjugate, 149
- step\_eval, 148
- step\_evalf, 148
- step\_imag\_part, 149
- step\_real\_part, 149
- step\_series, 148
- structure< T, CP >::reg\_info, 285
- sub\_matrix, 208
- subs, 124
- subvalue, 198
- swap, 124, 129
- sy\_anti, 269, 270
- sy\_cycl, 270

- [sy\\_none](#), [268](#)
- [sy\\_symm](#), [269](#)
- [sym\\_desc\\_vec](#), [92](#)
- [symbolic\\_matrix](#), [207](#), [210](#)
- [symm](#), [267](#)
- [symmetric2](#), [266](#)
- [symmetric3](#), [266](#)
- [symmetric4](#), [266](#)
- [symmetrize](#), [122](#), [267](#), [271](#)
- [symmetrize\\_cyclic](#), [123](#), [268](#), [271](#)
- [synthesize\\_func](#), [68](#)
- [tan](#), [231](#)
- [tan\\_conjugate](#), [185](#)
- [tan\\_deriv](#), [185](#)
- [tan\\_eval](#), [184](#)
- [tan\\_evalf](#), [184](#)
- [tan\\_imag\\_part](#), [185](#)
- [tan\\_real\\_part](#), [185](#)
- [tan\\_series](#), [185](#)
- [tanh](#), [234](#)
- [tanh\\_conjugate](#), [194](#)
- [tanh\\_deriv](#), [194](#)
- [tanh\\_eval](#), [193](#)
- [tanh\\_evalf](#), [193](#)
- [tanh\\_imag\\_part](#), [194](#)
- [tanh\\_real\\_part](#), [194](#)
- [tanh\\_series](#), [194](#)
- [tensor](#), [283](#)
- [tgamma](#), [237](#)
- [tgamma\\_conjugate](#), [168](#)
- [tgamma\\_deriv](#), [168](#)
- [tgamma\\_eval](#), [167](#)
- [tgamma\\_evalf](#), [167](#)
- [tgamma\\_series](#), [168](#)
- [to\\_double](#), [248](#)
- [to\\_int](#), [248](#)
- [to\\_long](#), [248](#)
- [to\\_polynomial](#), [120](#)
- [to\\_rational](#), [120](#)
- [trace](#), [209](#)
- [trace\\_string](#), [102](#)
- [transpose](#), [209](#)
- [tree](#), [258](#)
- [trig\\_info](#), [182](#)
- [tryfactsubs](#), [211](#)
- [uintvector](#), [91](#)
- [unarch\\_table\\_instance](#), [282](#)
- [unarchive\\_map\\_t](#), [68](#)
- [unit\\_matrix](#), [207](#), [210](#)
- [unlink\\_ex](#), [129](#)
- [unsignedvector](#), [91](#)
- [version\\_major](#), [286](#)
- [version\\_micro](#), [286](#)
- [version\\_minor](#), [286](#)
- [wild](#), [282](#)
- [write\\_real\\_float](#), [228](#)
- [write\\_unsigned](#), [93](#)
- [zeta](#), [160](#), [236](#)
- [zeta1\\_deriv](#), [176](#)
- [zeta1\\_eval](#), [176](#)
- [zeta1\\_evalf](#), [176](#)
- [zeta1\\_print\\_latex](#), [176](#)
- [zeta2\\_deriv](#), [177](#)
- [zeta2\\_eval](#), [177](#)
- [zeta2\\_evalf](#), [177](#)
- [zeta2\\_print\\_latex](#), [177](#)
- [zetaderiv\\_deriv](#), [154](#)
- [zetaderiv\\_eval](#), [154](#)
- [ginac Directory Reference](#), [23](#)
- [ginac.h](#), [1290](#)
- [GiNaC::numeric\\_digits](#), [307](#)
  - [\\_numeric\\_digits](#), [308](#)
  - [add\\_callback](#), [308](#)
  - [callbacklist](#), [309](#)
  - [digits](#), [309](#)
  - [operator long](#), [308](#)
  - [operator=](#), [308](#)
  - [print](#), [308](#)
  - [too\\_late](#), [309](#)
- [GiNaC::add](#), [310](#)
  - [add](#), [316](#), [317](#)
  - [coeff](#), [318](#)
  - [combine\\_ex\\_with\\_coeff\\_to\\_pair](#), [324](#)
  - [combine\\_pair\\_with\\_coeff\\_to\\_pair](#), [324](#)
  - [conjugate](#), [321](#)
  - [degree](#), [318](#)
  - [derivative](#), [322](#)
  - [do\\_print](#), [325](#)
  - [do\\_print\\_csrc](#), [325](#)
  - [do\\_print\\_latex](#), [325](#)
  - [do\\_print\\_python\\_repr](#), [326](#)
  - [eval](#), [319](#)
  - [eval\\_ncmul](#), [322](#)
  - [evalm](#), [319](#)
  - [expand](#), [324](#)
  - [get\\_free\\_indices](#), [322](#)
  - [imag\\_part](#), [322](#)
  - [info](#), [317](#)
  - [integer\\_content](#), [320](#)
  - [is\\_polynomial](#), [318](#)
  - [ldegree](#), [318](#)
  - [max\\_coefficient](#), [321](#)
  - [mul](#), [326](#)
  - [normal](#), [320](#)
  - [power](#), [326](#)
  - [precedence](#), [317](#)
  - [print\\_add](#), [325](#)
  - [real\\_part](#), [321](#)
  - [recombine\\_pair\\_to\\_ex](#), [324](#)
  - [return\\_type](#), [322](#)
  - [return\\_type\\_tinfo](#), [323](#)
  - [series](#), [319](#)
  - [smod](#), [320](#)
  - [split\\_ex\\_to\\_pair](#), [323](#)
  - [thisexpairseq](#), [323](#)
- [GiNaC::archive](#), [327](#)

- [~archive](#), [328](#)
  - [add\\_node](#), [331](#)
  - [archive](#), [328](#), [329](#)
  - [archive\\_ex](#), [329](#)
  - [atomize](#), [332](#)
  - [atoms](#), [333](#)
  - [clear](#), [331](#)
  - [exprs](#), [333](#)
  - [exprtable](#), [333](#)
  - [forget](#), [331](#)
  - [get\\_node](#), [331](#)
  - [get\\_top\\_node](#), [331](#)
  - [inv\\_at\\_cit](#), [328](#)
  - [inverse\\_atoms](#), [333](#)
  - [nodes](#), [333](#)
  - [num\\_expressions](#), [330](#)
  - [operator<<](#), [332](#)
  - [operator>>](#), [332](#)
  - [printraw](#), [332](#)
  - [unarchive\\_ex](#), [329](#), [330](#)
  - [unatomize](#), [332](#)
- [GiNaC::archive::archived\\_ex](#), [345](#)
  - [archived\\_ex](#), [345](#)
  - [name](#), [346](#)
  - [root](#), [346](#)
- [GiNaC::archive\\_node](#), [334](#)
  - [a](#), [343](#)
  - [add\\_bool](#), [337](#)
  - [add\\_ex](#), [338](#)
  - [add\\_string](#), [338](#)
  - [add\\_unsigned](#), [337](#)
  - [archive\\_node](#), [337](#)
  - [archive\\_node\\_cit](#), [336](#)
  - [e](#), [343](#)
  - [find\\_bool](#), [338](#)
  - [find\\_ex](#), [340](#)
  - [find\\_ex\\_by\\_loc](#), [340](#)
  - [find\\_ex\\_node](#), [340](#)
  - [find\\_first](#), [339](#)
  - [find\\_last](#), [339](#)
  - [find\\_property\\_range](#), [339](#)
  - [find\\_string](#), [339](#)
  - [find\\_unsigned](#), [338](#)
  - [forget](#), [342](#)
  - [get\\_ex](#), [341](#)
  - [get\\_properties](#), [341](#)
  - [has\\_ex](#), [341](#)
  - [has\\_expression](#), [343](#)
  - [has\\_same\\_ex\\_as](#), [341](#)
  - [operator<<](#), [342](#)
  - [operator>>](#), [342](#)
  - [operator=](#), [337](#)
  - [printraw](#), [342](#)
  - [property\\_type](#), [336](#)
  - [propinfovector](#), [336](#)
  - [props](#), [343](#)
  - [PTYPE\\_BOOL](#), [336](#)
  - [PTYPE\\_NODE](#), [337](#)
  - [PTYPE\\_STRING](#), [337](#)
  - [PTYPE\\_UNSIGNED](#), [337](#)
  - [unarchive](#), [341](#)
- [GiNaC::archive\\_node::archive\\_node\\_cit\\_range](#), [344](#)
  - [begin](#), [345](#)
  - [end](#), [345](#)
- [GiNaC::archive\\_node::property](#), [1027](#)
  - [name](#), [1028](#)
  - [property](#), [1028](#)
  - [type](#), [1028](#)
  - [value](#), [1028](#)
- [GiNaC::archive\\_node::property\\_info](#), [1029](#)
  - [count](#), [1030](#)
  - [name](#), [1030](#)
  - [property\\_info](#), [1029](#)
  - [type](#), [1030](#)
- [GiNaC::basic](#), [346](#)
  - [~basic](#), [351](#)
  - [accept](#), [358](#)
  - [add\\_indexed](#), [364](#)
  - [archive](#), [368](#)
  - [basic](#), [351](#)
  - [calchash](#), [367](#)
  - [clearflag](#), [372](#)
  - [coeff](#), [359](#)
  - [collect](#), [359](#)
  - [compare](#), [370](#)
  - [compare\\_same\\_type](#), [367](#)
  - [conjugate](#), [366](#)
  - [contract\\_with](#), [365](#)
  - [dbgprint](#), [354](#)
  - [dbgprinttree](#), [354](#)
  - [degree](#), [358](#)
  - [derivative](#), [361](#)
  - [diff](#), [370](#)
  - [do\\_print](#), [373](#)
  - [do\\_print\\_python\\_repr](#), [373](#)
  - [do\\_print\\_tree](#), [373](#)
  - [duplicate](#), [352](#)
  - [ensure\\_if\\_modifiable](#), [372](#)
  - [eval](#), [352](#)
  - [eval\\_indexed](#), [353](#)
  - [eval\\_integ](#), [353](#)
  - [eval\\_ncmul](#), [353](#)
  - [evalf](#), [352](#)
  - [evalm](#), [352](#)
  - [ex](#), [374](#)
  - [expand](#), [359](#)
  - [flags](#), [375](#)
  - [get\\_free\\_indices](#), [363](#)
  - [gethash](#), [371](#)
  - [has](#), [356](#)
  - [hashvalue](#), [375](#)
  - [hold](#), [371](#)
  - [imag\\_part](#), [366](#)
  - [info](#), [354](#)
  - [integer\\_content](#), [362](#)
  - [is\\_equal](#), [370](#)

- is\_equal\_same\_type, 367
- is\_polynomial, 358
- ldegree, 359
- let\_op, 356
- map, 358
- match, 357
- match\_same\_type, 357
- max\_coefficient, 363
- nops, 355
- normal, 361
- op, 355
- operator=, 352
- operator[], 355, 356
- precedence, 354
- print, 353
- print\_dispatch, 368
- read\_archive, 369
- real\_part, 366
- return\_type, 365
- return\_type\_tinfo, 366
- scalar\_mul\_indexed, 364
- series, 361
- setflag, 372
- smod, 363
- subs, 357
- subs\_one\_level, 369
- to\_polynomial, 362
- to\_rational, 362
- GiNaC::basic\_log\_kernel, 376
  - do\_print, 381
  - series\_coeff\_impl, 381
- GiNaC::basic\_multi\_iterator< T >, 382
  - ~basic\_multi\_iterator, 384
  - B, 387
  - basic\_multi\_iterator, 384
  - flag\_overflow, 388
  - get\_vector, 385
  - init, 386
  - N, 387
  - operator<<, 387
  - operator(), 386
  - operator++, 386
  - operator[], 385, 386
  - overflow, 385
  - size, 385
  - v, 387
- GiNaC::basic\_partition\_generator, 388
  - basic\_partition\_generator, 390
  - mpgen, 390
- GiNaC::basic\_partition\_generator::mpartition2, 839
  - m, 840
  - mpartition2, 839
  - n, 840
  - next\_partition, 839
  - x, 840
- GiNaC::class\_info< OPT >, 390
  - class\_info, 392
  - dump\_hierarchy, 392
  - dump\_tree, 392
  - find, 392
  - first, 393
  - get\_parent, 392
  - identify\_parents, 393
  - next, 393
  - options, 393
  - parent, 393
  - parents\_identified, 393
- GiNaC::class\_info< OPT >::tree\_node, 1216
  - add\_child, 1217
  - children, 1217
  - info, 1217
  - tree\_node, 1217
- GiNaC::clifford, 394
  - archive, 402
  - clifford, 401, 402
  - commutator\_sign, 407
  - do\_print\_dflt, 406
  - do\_print\_latex, 407
  - do\_print\_tree, 407
  - eval\_ncmul, 403
  - get\_commutator\_sign, 405
  - get\_metric, 405
  - get\_representation\_label, 405
  - let\_op, 406
  - match\_same\_type, 403
  - metric, 407
  - nops, 405
  - op, 406
  - precedence, 402
  - read\_archive, 403
  - representation\_label, 407
  - return\_type, 404
  - return\_type\_tinfo, 404
  - same\_metric, 405
  - subs, 406
  - thiscontainer, 404
- GiNaC::cliffordunit, 408
  - contract\_with, 413
  - do\_print, 413
  - do\_print\_latex, 413
- GiNaC::color, 413
  - archive, 422
  - color, 421
  - eval\_ncmul, 422
  - get\_representation\_label, 424
  - match\_same\_type, 423
  - read\_archive, 422
  - representation\_label, 424
  - return\_type, 424
  - return\_type\_tinfo, 424
  - thiscontainer, 423
- GiNaC::compare\_all\_equal< T >, 425
  - ~compare\_all\_equal, 425
  - struct\_compare, 425
  - struct\_is\_equal, 425
- GiNaC::compare\_bitwise< T >, 426

- ~compare\_bitwise, 426
  - struct\_compare, 426
  - struct\_is\_equal, 426
- GiNaC::compare\_std\_less< T >, 427
  - ~compare\_std\_less, 427
  - struct\_compare, 427
  - struct\_is\_equal, 427
- GiNaC::composition\_generator, 428
  - atend, 429
  - cmgen, 429
  - composition, 430
  - composition\_generator, 429
  - current\_updated, 430
  - get, 429
  - next, 429
  - trivial, 429
- GiNaC::composition\_generator::coolmulti, 473
  - ~coolmulti, 473
  - after\_i, 474
  - coolmulti, 473
  - finished, 474
  - head, 474
  - i, 474
  - next\_permutation, 474
- GiNaC::composition\_generator::coolmulti::element, 533
  - ~element, 534
  - element, 534
  - next, 534
  - value, 534
- GiNaC::const\_iterator, 430
  - const\_iterator, 433
  - const\_postorder\_iterator, 436
  - const\_preorder\_iterator, 435
  - difference\_type, 432
  - e, 436
  - ex, 435
  - i, 436
  - iterator\_category, 432
  - operator!=, 435
  - operator<, 435
  - operator<=, 435
  - operator>, 435
  - operator>=, 435
  - operator+, 434, 436
  - operator++, 433
  - operator+=, 434
  - operator-, 434, 436
  - operator->, 433
  - operator--, 434
  - operator-=, 434
  - operator==, 434
  - operator[], 433
  - operator\*, 433
  - pointer, 432
  - reference, 432
  - value\_type, 432
- GiNaC::const\_postorder\_iterator, 437
  - const\_postorder\_iterator, 438
- descend, 439
  - difference\_type, 437
  - increment, 439
  - iterator\_category, 437
  - operator!=, 439
  - operator++, 438
  - operator->, 438
  - operator==, 439
  - operator\*, 438
  - pointer, 437
  - reference, 438
  - s, 439
  - value\_type, 437
- GiNaC::const\_preorder\_iterator, 440
  - const\_preorder\_iterator, 441
  - difference\_type, 440
  - increment, 442
  - iterator\_category, 440
  - operator!=, 442
  - operator++, 441
  - operator->, 441
  - operator==, 442
  - operator\*, 441
  - pointer, 440
  - reference, 441
  - s, 442
  - value\_type, 440
- GiNaC::constant, 443
  - archive, 450
  - calchash, 451
  - conjugate, 449
  - constant, 448
  - derivative, 450
  - do\_print, 452
  - do\_print\_latex, 452
  - do\_print\_python\_repr, 452
  - do\_print\_tree, 452
  - domain, 453
  - ef, 453
  - evalf, 449
  - get\_name, 451
  - get\_TeX\_name, 451
  - imag\_part, 450
  - info, 449
  - is\_equal\_same\_type, 451
  - is\_polynomial, 449
  - name, 453
  - next\_serial, 453
  - number, 453
  - read\_archive, 450
  - real\_part, 449
  - serial, 453
  - TeX\_name, 453
- GiNaC::container< C >, 454
  - append, 466
  - archive, 463
  - begin, 467
  - conjugate, 463

- const\_iterator, 460
- const\_reverse\_iterator, 460
- container, 460
- do\_print, 468
- do\_print\_python, 468
- do\_print\_python\_repr, 468
- do\_print\_tree, 468
- end, 467
- get\_close\_delim, 461
- get\_default\_flags, 461
- get\_open\_delim, 461
- imag\_part, 464
- info, 461
- is\_equal\_same\_type, 464
- let\_op, 462
- nops, 461
- op, 462
- precedence, 461
- prepend, 466
- printseq, 465
- rbegin, 468
- read\_archive, 463
- real\_part, 464
- remove\_all, 467
- remove\_first, 466
- remove\_last, 466
- rend, 468
- sort, 467
- sort\_, 465
- STLT, 460
- subs, 462
- subchildren, 469
- thiscontainer, 464, 465
- unique, 467
- unique\_, 466, 469
- GiNaC::container\_storage< C >, 469
  - ~container\_storage, 471
  - container\_storage, 470, 471
  - reserve, 471
  - seq, 472
  - STLT, 470
- GiNaC::derivative\_map\_function, 475
  - derivative\_map\_function, 476
  - operator(), 476
  - s, 476
- GiNaC::determinant\_algo, 477
  - automatic, 477
  - bareiss, 478
  - divfree, 477
  - gauss, 477
  - laplace, 477
- GiNaC::diracgamma, 478
  - contract\_with, 483
  - do\_print, 483
  - do\_print\_latex, 483
- GiNaC::diracgamma5, 484
  - conjugate, 488
  - do\_print, 488
  - do\_print\_latex, 488
- GiNaC::diracgammaL, 489
  - conjugate, 493
  - do\_print, 493
  - do\_print\_latex, 493
- GiNaC::diracgammaR, 494
  - conjugate, 498
  - do\_print, 498
  - do\_print\_latex, 498
- GiNaC::diracone, 499
  - do\_print, 503
  - do\_print\_latex, 503
- GiNaC::do\_taylor, 504
- GiNaC::domain, 504
  - complex, 504
  - positive, 504
  - real, 504
- GiNaC::dunno, 505
- GiNaC::Ebar\_kernel, 505
  - do\_print, 512
  - Ebar\_kernel, 511
  - get\_numerical\_value, 512
  - is\_numeric, 511
  - let\_op, 511
  - m, 513
  - n, 513
  - nops, 511
  - op, 511
  - series\_coeff\_impl, 512
  - x, 513
  - y, 513
- GiNaC::Eisenstein\_h\_kernel, 513
  - C\_norm, 523
  - coefficient\_a0, 522
  - coefficient\_an, 522
  - do\_print, 522
  - Eisenstein\_h\_kernel, 520
  - get\_numerical\_value, 521
  - is\_numeric, 521
  - k, 523
  - Laurent\_series, 521
  - let\_op, 520
  - N, 523
  - nops, 520
  - op, 520
  - q\_expansion\_modular\_form, 522
  - r, 523
  - s, 523
  - series, 520
  - uses\_Laurent\_series, 521
- GiNaC::Eisenstein\_kernel, 524
  - a, 533
  - b, 533
  - C\_norm, 533
  - do\_print, 532
  - Eisenstein\_kernel, 530
  - get\_numerical\_value, 531
  - is\_numeric, 531

- K, [533](#)
- k, [532](#)
- Laurent\_series, [531](#)
- let\_op, [531](#)
- N, [532](#)
- nops, [530](#)
- op, [531](#)
- q\_expansion\_modular\_form, [532](#)
- series, [530](#)
- uses\_Laurent\_series, [532](#)
- GiNaC::ELi\_kernel, [535](#)
  - do\_print, [542](#)
  - ELi\_kernel, [541](#)
  - get\_numerical\_value, [542](#)
  - is\_numeric, [541](#)
  - let\_op, [541](#)
  - m, [542](#)
  - n, [542](#)
  - nops, [541](#)
  - op, [541](#)
  - series\_coeff\_impl, [542](#)
  - x, [543](#)
  - y, [543](#)
- GiNaC::error\_and\_integral, [544](#)
  - error, [545](#)
  - error\_and\_integral, [545](#)
  - integral, [545](#)
- GiNaC::error\_and\_integral\_is\_less, [545](#)
  - operator(), [545](#)
- GiNaC::eval\_integ\_map\_function, [546](#)
  - operator(), [547](#)
- GiNaC::evalf\_map\_function, [547](#)
  - operator(), [548](#)
- GiNaC::evalm\_map\_function, [549](#)
  - operator(), [550](#)
- GiNaC::ex, [550](#)
  - accept, [564](#)
  - antisymmetrize, [576](#)
  - archive\_node, [580](#)
  - are\_ex\_trivially\_equal, [580](#)
  - begin, [556](#)
  - bp, [582](#)
  - coeff, [565](#)
  - collect, [566](#)
  - compare, [574](#)
  - conjugate, [561](#)
  - construct\_from\_basic, [577](#)
  - construct\_from\_double, [579](#)
  - construct\_from\_int, [577](#)
  - construct\_from\_long, [578](#)
  - construct\_from\_longlong, [578](#)
  - construct\_from\_string\_and\_lst, [579](#)
  - construct\_from\_uint, [578](#)
  - construct\_from\_ulong, [578](#)
  - construct\_from\_ulonglong, [579](#)
  - content, [570](#)
  - dbgprint, [558](#)
  - dbgprinttree, [558](#)
  - degree, [565](#)
  - denom, [569](#)
  - diff, [566](#)
  - end, [556](#)
  - eval, [557](#)
  - eval\_integ, [557](#)
  - eval\_ncmul, [557](#)
  - evalf, [557](#)
  - evalm, [557](#)
  - ex, [554](#), [555](#)
  - ex\_to, [580](#)
  - expand, [566](#)
  - find, [562](#)
  - get\_free\_indices, [573](#)
  - gethash, [577](#)
  - has, [562](#)
  - imag\_part, [562](#)
  - info, [558](#)
  - integer\_content, [571](#)
  - is\_a, [581](#)
  - is\_equal, [574](#)
  - is\_exactly\_a, [581](#)
  - is\_polynomial, [565](#)
  - is\_zero, [575](#)
  - is\_zero\_matrix, [575](#)
  - lcoeff, [565](#)
  - ldegree, [565](#)
  - let\_op, [560](#)
  - lhs, [561](#)
  - makewriteable, [579](#)
  - map, [563](#), [564](#)
  - match, [562](#)
  - max\_coefficient, [573](#)
  - nops, [559](#)
  - normal, [567](#)
  - numer, [569](#)
  - numer\_denom, [569](#)
  - op, [559](#)
  - operator[], [560](#), [561](#)
  - postorder\_begin, [557](#)
  - postorder\_end, [557](#)
  - preorder\_begin, [556](#)
  - preorder\_end, [556](#)
  - primpart, [571](#), [572](#)
  - print, [558](#)
  - real\_part, [561](#)
  - return\_type, [577](#)
  - return\_type\_tinfo, [577](#)
  - rhs, [561](#)
  - series, [567](#)
  - share, [579](#)
  - simplify\_indexed, [573](#), [574](#)
  - smod, [573](#)
  - subs, [563](#)
  - swap, [556](#)
  - symmetrize, [575](#), [576](#)
  - symmetrize\_cyclic, [576](#)
  - tcoeff, [566](#)



- to\_polynomial, 568
- to\_rational, 568
- traverse, 564
- traverse\_postorder, 564
- traverse\_preorder, 564
- unit, 570
- unitcontprim, 572
- GiNaC::ex\_base\_is\_less, 582
  - operator(), 582
- GiNaC::ex\_is\_equal, 582
  - operator(), 583
- GiNaC::ex\_is\_less, 583
  - operator(), 583
- GiNaC::ex\_swap, 583
  - operator(), 584
- GiNaC::expair, 584
  - coeff, 588
  - compare, 587
  - conjugate, 587
  - expair, 586
  - is\_canonical\_numeric, 587
  - is\_equal, 586
  - is\_less, 586
  - print, 587
  - rest, 588
  - swap, 587
- GiNaC::expair\_is\_less, 588
  - operator(), 589
- GiNaC::expair\_rest\_is\_less, 589
  - operator(), 589
- GiNaC::expair\_swap, 590
  - operator(), 590
- GiNaC::expairseq, 590
  - archive, 599
  - calchash, 600
  - can\_make\_flat, 603
  - canonicalize, 605
  - combine\_ex\_with\_coeff\_to\_pair, 602
  - combine\_overall\_coeff, 603
  - combine\_pair\_with\_coeff\_to\_pair, 602
  - combine\_same\_terms\_sorted\_seq, 606
  - conjugate, 598
  - construct\_from\_2\_ex, 604
  - construct\_from\_2\_expairseq, 604
  - construct\_from\_epvector, 605
  - construct\_from\_expairseq\_ex, 604
  - construct\_from\_exvector, 604
  - default\_overall\_coeff, 603
  - do\_print, 603
  - do\_print\_tree, 604
  - eval, 597
  - evalchildren, 606
  - expair\_needs\_further\_processing, 602
  - expairseq, 595, 596
  - expand, 600
  - expandchildren, 606
  - info, 596
  - is\_canonical, 606
  - is\_equal\_same\_type, 599
  - make\_flat, 605
  - map, 597
  - match, 598
  - nops, 596
  - op, 597
  - overall\_coeff, 607
  - precedence, 596
  - printpair, 601
  - printseq, 601
  - read\_archive, 599
  - recombine\_pair\_to\_ex, 602
  - return\_type, 600
  - seq, 607
  - split\_ex\_to\_pair, 601
  - subs, 598
  - subchildren, 607
  - thisexpairseq, 600, 601
  - to\_polynomial, 598
  - to\_rational, 597
- GiNaC::expand\_map\_function, 608
  - expand\_map\_function, 609
  - operator(), 610
  - options, 610
- GiNaC::expand\_options, 610
  - expand\_function\_args, 611
  - expand\_indexed, 610
  - expand\_rename\_idx, 611
  - expand\_transcendental, 611
- GiNaC::factor\_options, 611
  - all, 611
  - polynomial, 611
- GiNaC::fail, 612
  - do\_print, 616
  - return\_type, 616
- GiNaC::fderivative, 616
  - archive, 626
  - derivative, 627
  - derivatives, 628
  - do\_print, 628
  - do\_print\_csrc, 629
  - do\_print\_latex, 628
  - do\_print\_tree, 629
  - eval, 625
  - fderivative, 624, 625
  - is\_equal\_same\_type, 627
  - match\_same\_type, 627
  - parameter\_set, 629
  - print, 625
  - read\_archive, 626
  - series, 625
  - thiscontainer, 626
- GiNaC::function, 630
  - archive, 645
  - calchash, 643
  - conjugate, 644
  - current\_serial, 649
  - derivative, 645

- eval, 642
- eval\_ncmul, 643
- evalf, 642
- expand, 642
- expl\_derivative, 647
- find\_function, 648
- function, 637–641
- get\_name, 649
- get\_registered\_functions, 648
- get\_serial, 649
- imag\_part, 644
- info, 645
- is\_equal\_same\_type, 646
- lookup\_remember\_table, 647
- match\_same\_type, 646
- pderivative, 647
- power, 648
- precedence, 642
- print, 641
- read\_archive, 645
- real\_part, 644
- register\_new, 648
- registered\_functions, 647
- remember\_table\_entry, 649
- return\_type, 646
- return\_type\_tinfo, 647
- serial, 649
- series, 643
- store\_remember\_table, 648
- thiscontainer, 643, 644
- GiNaC::function\_options, 650
- ~function\_options, 657
- conjugate\_f, 687
- conjugate\_func, 662–664, 680
- conjugate\_use\_exvector\_args, 690
- derivative\_f, 688
- derivative\_func, 670–672, 681
- derivative\_use\_exvector\_args, 691
- do\_not\_evalf\_params, 684
- dummy, 657
- eval\_f, 687
- eval\_func, 658–660, 680
- eval\_use\_exvector\_args, 690
- evalf\_f, 687
- evalf\_func, 660–662, 680
- evalf\_params\_first, 689
- evalf\_use\_exvector\_args, 690
- expand\_f, 688
- expand\_func, 668–670, 681
- expand\_use\_exvector\_args, 691
- expl\_derivative\_f, 688
- expl\_derivative\_func, 672–674, 681
- expl\_derivative\_use\_exvector\_args, 691
- fderivative, 686
- function, 686
- function\_options, 656, 657
- functions\_with\_same\_name, 692
- get\_name, 685
- get\_nparams, 685
- has\_derivative, 685
- has\_power, 685
- imag\_part\_f, 688
- imag\_part\_func, 666–668, 681
- imag\_part\_use\_exvector\_args, 691
- info\_f, 689
- info\_func, 678–681
- info\_use\_exvector\_args, 691
- initialize, 657
- latex\_name, 658
- name, 687
- nparams, 687
- overloaded, 685
- power\_f, 688
- power\_func, 674–676, 681
- power\_use\_exvector\_args, 691
- print\_dispatch\_table, 689
- print\_func, 682–684
- print\_use\_exvector\_args, 691
- real\_part\_f, 688
- real\_part\_func, 664–666, 680
- real\_part\_use\_exvector\_args, 690
- remember, 684
- remember\_assoc\_size, 690
- remember\_size, 690
- remember\_strategy, 690
- return\_type, 689
- return\_type\_tinfo, 689
- series\_f, 689
- series\_func, 676–678, 681
- series\_use\_exvector\_args, 691
- set\_name, 658
- set\_print\_func, 686
- set\_return\_type, 684
- set\_symmetry, 685
- symtree, 692
- test\_and\_set\_nparams, 685
- TeX\_name, 687
- use\_remember, 690
- use\_return\_type, 689
- GiNaC::G2\_SERIAL, 692
- serial, 692
- GiNaC::G3\_SERIAL, 693
- serial, 693
- GiNaC::gcd\_options, 693
- no\_heur\_gcd, 693
- no\_part\_factored, 694
- use\_sr\_gcd, 694
- GiNaC::gcdheu\_failed, 694
- GiNaC::Gt, 694
- apply\_function\_recursive, 707
- applyIntegrationPath, 704
- archive, 702
- args, 709
- auto\_clear\_polylog\_cache, 709
- calchash, 701
- clear\_polylog\_cache, 707

- cutIntegrationPath\_threshold\_imag, 709
- cutIntegrationPath\_threshold\_real, 709
- decomposeIntegrationPath, 703
- do\_print, 708
- do\_print\_latex, 708
- enable\_tauToFundamental, 709
- eval, 700
- evalf, 701
- evaluate, 704
- ex\_cutIntegrationPath, 705, 706
- ex\_evaluate, 705, 707
- ex\_prepare, 705, 707
- ex\_qExpand, 705, 707
- ex\_regularise, 704, 706
- ex\_tauToFundamental, 705, 706
- ex\_zisToFundamental, 704, 706
- findMoebiusTransform, 703
- getArgs, 700
- getTau, 700
- getZ, 700
- Gt, 699
- has, 702
- lst\_evaluate, 706, 707
- match, 702
- nargs, 709
- nops, 701
- nqexand\_transformer, 710
- precedence, 700
- qExpand, 704
- qExpand\_minOrder, 709
- qExpand\_stepSize, 709
- read\_archive, 702
- regularise, 703
- setArgs, 700
- subs, 701
- tau, 710
- tauToFundamental, 703
- to\_integer, 708
- to\_numeric, 708
- z, 710
- zisToFundamental, 703
- GiNaC::Gt::kernel, 770
  - check, 772
  - deform, 773
  - Gt, 772
  - kernel, 772
  - ni, 773
  - zi, 773
- GiNaC::Gt\_detail, 300
  - deconcatenate\_path, 301
  - deconcatenate\_path< ex >, 301
  - deconcatenate\_path< Gt::kernel >, 301
  - integer\_partition, 301
  - operator<, 302
  - operator<<, 302
- GiNaC::Gt\_detail::pathintegral\_term, 946
  - denom, 949
  - evaluate, 948
  - G\_path, 948
  - integrate, 948
  - operator<, 949
  - operator<<, 948
  - pathintegral\_term, 947
  - pathintegral\_term\_list, 948
  - polylog, 949
  - power, 949
  - prefactor, 949
- GiNaC::Gt\_detail::pathintegral\_term\_list, 950
  - add, 951
  - begin, 950
  - clear, 950
  - empty, 950
  - end, 950
  - pop, 951
  - size, 951
  - terms, 952
- GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, 1213
  - cache, 1216
  - check\_type, 1215
  - clear\_cache, 1215
  - func\_default, 1216
  - func\_obj, 1216
  - get\_cache, 1215
  - impl, 1215
  - operator(), 1214
  - print\_cache, 1214
  - TransformExpressionWithCache, 1214
- GiNaC::has\_distance< T >, 710
  - no\_type, 711
  - test, 712
  - value, 711
  - yes\_type, 711
- GiNaC::has\_options, 712
  - algebraic, 712
- GiNaC::idx, 713
  - archive, 721
  - calchash, 722
  - derivative, 722
  - dim, 726
  - do\_print, 725
  - do\_print\_csrc, 725
  - do\_print\_latex, 725
  - do\_print\_tree, 725
  - evalf, 720
  - get\_dim, 724
  - get\_value, 723
  - idx, 719
  - info, 720
  - is\_dim\_numeric, 724
  - is\_dim\_symbolic, 724
  - is\_dummy\_pair\_same\_type, 723
  - is\_numeric, 723
  - is\_symbolic, 723
  - map, 720
  - match\_same\_type, 722

- minimal\_dim, [724](#)
- nops, [720](#)
- op, [720](#)
- print\_index, [725](#)
- read\_archive, [721](#)
- replace\_dim, [724](#)
- subs, [721](#)
- value, [726](#)
- GiNaC::idx\_is\_equal\_ignore\_dim, [726](#)
- operator(), [727](#)
- GiNaC::indexed, [727](#)
  - all\_index\_values\_are, [742](#)
  - archive, [740](#)
  - derivative, [741](#)
  - do\_print, [744](#)
  - do\_print\_latex, [744](#)
  - do\_print\_tree, [744](#)
  - eval, [739](#)
  - expand, [742](#)
  - get\_dummy\_indices, [742](#), [743](#)
  - get\_free\_indices, [740](#)
  - get\_indices, [742](#)
  - get\_symmetry, [743](#)
  - has\_dummy\_index\_for, [743](#)
  - imag\_part, [740](#)
  - indexed, [734–736](#), [738](#), [739](#)
  - info, [739](#)
  - precedence, [739](#)
  - print\_indexed, [743](#)
  - printindices, [743](#)
  - read\_archive, [740](#)
  - real\_part, [740](#)
  - reposition\_dummy\_indices, [745](#)
  - return\_type, [741](#)
  - return\_type\_tinfo, [742](#)
  - simplify\_indexed, [745](#)
  - simplify\_indexed\_product, [745](#)
  - symtree, [746](#)
  - thiscontainer, [741](#)
  - validate, [744](#)
- GiNaC::info\_flags, [746](#)
  - cinteger, [747](#)
  - cinteger\_polynomial, [747](#)
  - crational, [747](#)
  - crational\_polynomial, [747](#)
  - even, [747](#)
  - expanded, [747](#)
  - exprseq, [747](#)
  - has\_indices, [747](#)
  - idx, [747](#)
  - indefinite, [747](#)
  - indexed, [747](#)
  - integer, [747](#)
  - integer\_polynomial, [747](#)
  - list, [747](#)
  - negative, [747](#)
  - negint, [747](#)
  - nonnegative, [747](#)
  - nonnegint, [747](#)
  - numeric, [746](#)
  - odd, [747](#)
  - polynomial, [747](#)
  - posint, [747](#)
  - positive, [747](#)
  - prime, [747](#)
  - rational, [747](#)
  - rational\_function, [747](#)
  - rational\_polynomial, [747](#)
  - real, [747](#)
  - relation, [747](#)
  - relation\_equal, [747](#)
  - relation\_greater, [747](#)
  - relation\_greater\_or\_equal, [747](#)
  - relation\_less, [747](#)
  - relation\_less\_or\_equal, [747](#)
  - relation\_not\_equal, [747](#)
  - symbol, [747](#)
- GiNaC::integral, [747](#)
  - a, [758](#)
  - archive, [756](#)
  - b, [758](#)
  - conjugate, [756](#)
  - degree, [754](#)
  - derivative, [757](#)
  - do\_print, [757](#)
  - do\_print\_latex, [758](#)
  - eval, [753](#)
  - eval\_integ, [756](#)
  - eval\_ncmul, [754](#)
  - evalf, [753](#)
  - expand, [755](#)
  - f, [759](#)
  - get\_free\_indices, [755](#)
  - integral, [753](#)
  - ldegree, [754](#)
  - let\_op, [755](#)
  - max\_integration\_level, [758](#)
  - nops, [754](#)
  - op, [754](#)
  - precedence, [753](#)
  - read\_archive, [756](#)
  - relative\_integration\_error, [758](#)
  - return\_type, [755](#)
  - return\_type\_tinfo, [756](#)
  - series, [757](#)
  - x, [758](#)
- GiNaC::integration\_kernel, [759](#)
  - cache\_step\_size, [768](#)
  - do\_print, [767](#)
  - get\_cache\_size, [766](#)
  - get\_numerical\_value, [765](#)
  - get\_numerical\_value\_impl, [767](#)
  - get\_series\_coeff, [766](#)
  - has\_trailing\_zero, [764](#)
  - is\_numeric, [764](#)
  - Laurent\_series, [765](#)

- series, 764
- series\_coeff, 767
- series\_coeff\_impl, 766
- series\_vec, 768
- set\_cache\_step, 766
- uses\_Laurent\_series, 765
- GiNaC::internal, 302
- GiNaC::internal::\_iter\_rep, 305
- \_iter\_rep, 306
- e, 306
- i, 306
- i\_end, 307
- operator!=, 306
- operator==, 306
- GiNaC::is\_not\_a\_clifford, 768
- operator(), 768
- GiNaC::is\_summation\_idx, 769
- operator(), 769
- GiNaC::iterated\_integral2\_SERIAL, 769
- serial, 769
- GiNaC::iterated\_integral3\_SERIAL, 770
- serial, 770
- GiNaC::Kronecker\_dtau\_kernel, 773
- C\_norm, 781
- do\_print, 780
- get\_numerical\_value, 780
- is\_numeric, 779
- K, 781
- Kronecker\_dtau\_kernel, 779
- let\_op, 779
- n, 781
- nops, 779
- op, 779
- series\_coeff\_impl, 780
- z, 781
- GiNaC::Kronecker\_dz\_kernel, 782
- C\_norm, 790
- do\_print, 789
- get\_numerical\_value, 789
- is\_numeric, 788
- K, 790
- Kronecker\_dz\_kernel, 788
- let\_op, 788
- n, 790
- nops, 788
- op, 788
- series\_coeff\_impl, 789
- tau, 790
- z\_j, 790
- GiNaC::lanczos\_coeffs, 790
- calc\_lanczos\_A, 791
- coeffs, 792
- current\_vector, 792
- get\_order, 791
- lanczos\_coeffs, 791
- sufficiently\_accurate, 791
- GiNaC::library\_init, 793
- ~library\_init, 794
- count, 795
- init\_unarchivers, 794
- library\_init, 794
- GiNaC::make\_flat\_inserter, 795
- combine\_indices, 796
- do\_renaming, 796
- handle\_factor, 796
- make\_flat\_inserter, 796
- used\_indices, 796
- GiNaC::map\_function, 797
- ~map\_function, 799
- argument\_type, 799
- operator(), 799
- result\_type, 799
- GiNaC::matrix, 800
- add, 811
- add\_indexed, 808
- archive, 809
- charpoly, 815
- col, 822
- cols, 810
- conjugate, 809
- contract\_with, 808
- determinant, 814
- determinant\_minor, 818
- division\_free\_elimination, 819
- do\_print, 821
- do\_print\_latex, 821
- do\_print\_python\_repr, 821
- echelon\_form, 818
- eval\_indexed, 808
- evalm, 807
- fraction\_free\_elimination, 819
- gauss\_elimination, 818
- imag\_part, 809
- inverse, 816
- is\_zero\_matrix, 818
- let\_op, 807
- m, 822
- markowitz\_elimination, 820
- match\_same\_type, 810
- matrix, 805, 806
- mul, 811
- mul\_scalar, 812
- nops, 807
- op, 807
- operator(), 812
- pivot, 820
- pow, 812
- print\_elements, 821
- rank, 817
- read\_archive, 809
- real\_part, 809
- return\_type, 810
- row, 822
- rows, 810
- scalar\_mul\_indexed, 808
- set, 814

- solve, 817
- sub, 811
- subs, 807
- trace, 815
- transpose, 814
- GiNaC::minkmetric, 823
  - archive, 829
  - do\_print, 829
  - do\_print\_latex, 829
  - eval\_indexed, 828
  - info, 828
  - minkmetric, 828
  - pos\_sig, 830
  - read\_archive, 829
  - return\_type, 829
- GiNaC::modular\_form\_kernel, 830
  - C\_norm, 838
  - do\_print, 838
  - get\_numerical\_value, 837
  - is\_numeric, 837
  - k, 838
  - Laurent\_series, 837
  - let\_op, 836
  - modular\_form\_kernel, 836
  - nops, 836
  - op, 836
  - P, 838
  - q\_expansion\_modular\_form, 838
  - series, 836
  - uses\_Laurent\_series, 837
- GiNaC::mul, 840
  - add, 858
  - algebraic\_subs\_mul, 856
  - can\_be\_further\_expanded, 857
  - can\_make\_flat, 855
  - coeff, 848
  - combine\_ex\_with\_coeff\_to\_pair, 854
  - combine\_overall\_coeff, 855
  - combine\_pair\_with\_coeff\_to\_pair, 854
  - conjugate, 852
  - default\_overall\_coeff, 855
  - degree, 848
  - derivative, 852
  - do\_print, 856
  - do\_print\_csrc, 857
  - do\_print\_latex, 857
  - do\_print\_python\_repr, 857
  - eval, 849
  - eval\_ncmul, 852
  - evalf, 849
  - evalm, 850
  - expair\_needs\_further\_processing, 854
  - expand, 855
  - expandchildren, 857
  - find\_real\_imag, 856
  - get\_free\_indices, 852
  - has, 849
  - imag\_part, 850
  - info, 847
  - integer\_content, 851
  - is\_polynomial, 848
  - ldegree, 848
  - max\_coefficient, 851
  - mul, 846, 847
  - ncmul, 858
  - normal, 850
  - power, 858
  - precedence, 847
  - print\_overall\_coeff, 856
  - real\_part, 850
  - recombine\_pair\_to\_ex, 854
  - return\_type, 853
  - return\_type\_tinfo, 853
  - series, 850
  - smod, 851
  - split\_ex\_to\_pair, 853
  - thisexpairseq, 853
- GiNaC::multi\_iterator\_counter< T >, 859
  - init, 861
  - multi\_iterator\_counter, 861
  - operator<<, 862
  - operator++, 861
- GiNaC::multi\_iterator\_counter\_indv< T >, 862
  - init, 865
  - multi\_iterator\_counter\_indv, 864, 865
  - Nv, 866
  - operator<<, 866
  - operator++, 865
- GiNaC::multi\_iterator\_ordered< T >, 866
  - init, 869
  - multi\_iterator\_ordered, 868, 869
  - operator<<, 870
  - operator++, 869
- GiNaC::multi\_iterator\_ordered\_eq< T >, 870
  - init, 873
  - multi\_iterator\_ordered\_eq, 872
  - operator<<, 873
  - operator++, 873
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 874
  - init, 876
  - multi\_iterator\_ordered\_eq\_indv, 876
  - Nv, 877
  - operator<<, 877
  - operator++, 876
- GiNaC::multi\_iterator\_permutation< T >, 878
  - get\_sign, 881
  - init, 881
  - multi\_iterator\_permutation, 880
  - operator<<, 882
  - operator++, 881
- GiNaC::multi\_iterator\_shuffle< T >, 882
  - init, 885
  - multi\_iterator\_shuffle, 884
  - N\_internal, 885
  - operator<<, 885
  - operator++, 885

- `v_internal`, 885
  - `v_orig`, 886
- `GiNaC::multi_iterator_shuffle_prime< T >`, 886
  - `init`, 889
  - `multi_iterator_shuffle_prime`, 889
  - `operator<<`, 889
- `GiNaC::multiple_polylog_kernel`, 890
  - `do_print`, 897
  - `is_numeric`, 896
  - `let_op`, 896
  - `multiple_polylog_kernel`, 896
  - `nops`, 896
  - `op`, 896
  - `series_coeff_impl`, 896
  - `z`, 897
- `GiNaC::ncmul`, 898
  - `append_factors`, 909
  - `coeff`, 906
  - `conjugate`, 907
  - `count_factors`, 909
  - `degree`, 905
  - `derivative`, 908
  - `do_print`, 909
  - `do_print_csrc`, 909
  - `eval`, 906
  - `evalm`, 906
  - `expand`, 905
  - `expandchildren`, 909
  - `get_factors`, 910
  - `get_free_indices`, 907
  - `hold_ncmul`, 910
  - `imag_part`, 908
  - `info`, 905
  - `ldegree`, 905
  - `ncmul`, 904
  - `power`, 910
  - `precedence`, 905
  - `real_part`, 908
  - `reeval_ncmul`, 910
  - `return_type`, 908
  - `return_type_tinfo`, 908
  - `thiscontainer`, 907
- `GiNaC::normal_map_function`, 911
  - `operator()`, 912
- `GiNaC::numeric`, 912
  - `add`, 926
  - `add_dyn`, 928
  - `archive`, 925
  - `calchash`, 926
  - `coeff`, 921
  - `compare`, 931
  - `conjugate`, 925
  - `csgn`, 930
  - `degree`, 921
  - `denom`, 937
  - `derivative`, 926
  - `div`, 927
  - `div_dyn`, 928
  - `do_print`, 938
  - `do_print_csrc`, 939
  - `do_print_csrc_cl_N`, 939
  - `do_print_latex`, 938
  - `do_print_python_repr`, 939
  - `do_print_tree`, 939
  - `eval`, 922
  - `evalf`, 922
  - `has`, 922
  - `imag`, 937
  - `imag_part`, 925
  - `info`, 920
  - `int_length`, 938
  - `integer_content`, 924
  - `inverse`, 930
  - `is_cinteger`, 934
  - `is_crational`, 934
  - `is_equal`, 931
  - `is_equal_same_type`, 926
  - `is_even`, 933
  - `is_integer`, 932
  - `is_negative`, 932
  - `is_nonneg_integer`, 933
  - `is_odd`, 933
  - `is_polynomial`, 921
  - `is_pos_integer`, 932
  - `is_positive`, 932
  - `is_prime`, 933
  - `is_rational`, 933
  - `is_real`, 934
  - `is_zero`, 931
  - `ldegree`, 921
  - `max_coefficient`, 924
  - `mul`, 927
  - `mul_dyn`, 928
  - `normal`, 923
  - `numer`, 937
  - `numeric`, 919, 920
  - `operator!=`, 934
  - `operator<`, 935
  - `operator<=`, 935
  - `operator>`, 935
  - `operator>=`, 935
  - `operator=`, 929, 930
  - `operator==`, 934
  - `power`, 927
  - `power_dyn`, 929
  - `precedence`, 920
  - `print_numeric`, 938
  - `read_archive`, 925
  - `real`, 937
  - `real_part`, 925
  - `smod`, 924
  - `step`, 930
  - `sub`, 927
  - `sub_dyn`, 928
  - `subs`, 922
  - `to_cl_N`, 936

- to\_double, [936](#)
- to\_int, [936](#)
- to\_long, [936](#)
- to\_polynomial, [923](#)
- to\_rational, [923](#)
- value, [940](#)
- GiNaC::op0\_is\_equal, [940](#)
- operator(), [940](#)
- GiNaC::partition\_generator, [941](#)
  - current\_updated, [943](#)
  - get, [942](#)
  - next, [942](#)
  - partition, [943](#)
  - partition\_generator, [942](#)
- GiNaC::partition\_with\_zero\_parts\_generator, [943](#)
  - current\_updated, [945](#)
  - get, [945](#)
  - m, [945](#)
  - next, [945](#)
  - partition, [945](#)
  - partition\_with\_zero\_parts\_generator, [945](#)
- GiNaC::pointer\_to\_map\_function, [952](#)
  - operator(), [954](#)
  - pointer\_to\_map\_function, [954](#)
  - ptr, [954](#)
- GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [954](#)
  - arg1, [956](#)
  - operator(), [956](#)
  - pointer\_to\_map\_function\_1arg, [956](#)
  - ptr, [956](#)
- GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [957](#)
  - arg1, [959](#)
  - arg2, [959](#)
  - operator(), [958](#)
  - pointer\_to\_map\_function\_2args, [958](#)
  - ptr, [959](#)
- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [959](#)
  - arg1, [961](#)
  - arg2, [961](#)
  - arg3, [962](#)
  - operator(), [961](#)
  - pointer\_to\_map\_function\_3args, [961](#)
  - ptr, [961](#)
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, [962](#)
  - c, [964](#)
  - operator(), [964](#)
  - pointer\_to\_member\_to\_map\_function, [964](#)
  - ptr, [964](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [965](#)
  - arg1, [967](#)
  - c, [967](#)
  - operator(), [967](#)
  - pointer\_to\_member\_to\_map\_function\_1arg, [967](#)
  - ptr, [967](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [968](#)
  - arg1, [970](#)
  - arg2, [971](#)
  - c, [970](#)
  - operator(), [970](#)
  - pointer\_to\_member\_to\_map\_function\_2args, [970](#)
  - ptr, [970](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [971](#)
  - arg1, [973](#)
  - arg2, [974](#)
  - arg3, [974](#)
  - c, [973](#)
  - operator(), [973](#)
  - pointer\_to\_member\_to\_map\_function\_3args, [973](#)
  - ptr, [973](#)
- GiNaC::pole\_error, [974](#)
  - deg, [976](#)
  - degree, [975](#)
  - pole\_error, [975](#)
- GiNaC::possymbol, [976](#)
  - duplicate, [982](#)
  - get\_domain, [982](#)
  - possymbol, [981](#)
- GiNaC::power, [982](#)
  - archive, [993](#)
  - basis, [998](#)
  - coeff, [989](#)
  - conjugate, [993](#)
  - degree, [989](#)
  - derivative, [994](#)
  - do\_print\_csrc, [996](#)
  - do\_print\_csrc\_cl\_N, [996](#)
  - do\_print\_dflt, [995](#)
  - do\_print\_latex, [995](#)
  - do\_print\_python, [996](#)
  - do\_print\_python\_repr, [996](#)
  - eval, [989](#)
  - eval\_ncmul, [994](#)
  - evalf, [990](#)
  - evalm, [990](#)
  - expand, [995](#)
  - expand\_add, [996](#)
  - expand\_add\_2, [997](#)
  - expand\_mul, [997](#)
  - exponent, [998](#)
  - has, [991](#)
  - imag\_part, [993](#)
  - info, [987](#)
  - is\_polynomial, [988](#)
  - ldegree, [989](#)
  - map, [988](#)
  - mul, [998](#)
  - nops, [988](#)
  - normal, [992](#)
  - op, [988](#)
  - power, [987](#)



- precedence, 987
- print\_power, 995
- read\_archive, 993
- real\_part, 993
- return\_type, 994
- return\_type\_tinfo, 994
- series, 991
- subs, 991
- to\_polynomial, 992
- to\_rational, 992
- GiNaC::print\_context, 999
  - ~print\_context, 999
  - options, 1000
  - print\_context, 999
  - s, 1000
- GiNaC::print\_context\_options, 1001
  - get\_id, 1002
  - get\_name, 1001
  - get\_parent\_name, 1001
  - id, 1002
  - name, 1002
  - parent\_name, 1002
  - print\_context\_options, 1001
- GiNaC::print\_csrc, 1003
  - print\_csrc, 1004
- GiNaC::print\_csrc\_cl\_N, 1004
  - print\_csrc\_cl\_N, 1006
- GiNaC::print\_csrc\_double, 1006
  - print\_csrc\_double, 1008
- GiNaC::print\_csrc\_float, 1008
  - print\_csrc\_float, 1010
- GiNaC::print\_dflt, 1010
  - print\_dflt, 1011
- GiNaC::print\_functor, 1011
  - impl, 1013
  - is\_valid, 1013
  - operator(), 1013
  - operator=, 1013
  - print\_functor, 1012, 1013
- GiNaC::print\_functor\_impl, 1014
  - ~print\_functor\_impl, 1014
  - duplicate, 1014
  - operator(), 1014
- GiNaC::print\_latex, 1015
  - print\_latex, 1016
- GiNaC::print\_memfun\_handler< T, C >, 1017
  - duplicate, 1018
  - F, 1018
  - f, 1019
  - operator(), 1018
  - print\_memfun\_handler, 1018
- GiNaC::print\_options, 1019
  - print\_index\_dimensions, 1019
- GiNaC::print\_ptrfun\_handler< T, C >, 1020
  - duplicate, 1021
  - F, 1021
  - f, 1022
  - operator(), 1021
  - print\_ptrfun\_handler, 1021
- GiNaC::print\_python, 1022
  - print\_python, 1023
- GiNaC::print\_python\_repr, 1024
  - print\_python\_repr, 1025
- GiNaC::print\_tree, 1025
  - delta\_indent, 1027
  - print\_tree, 1026
- GiNaC::pseries, 1030
  - add\_series, 1043
  - archive, 1040
  - coeff, 1037
  - coeffop, 1042
  - collect, 1037
  - conjugate, 1039
  - convert\_to\_poly, 1042
  - degree, 1036
  - derivative, 1041
  - do\_print, 1045
  - do\_print\_latex, 1045
  - do\_print\_python, 1045
  - do\_print\_python\_repr, 1046
  - do\_print\_tree, 1045
  - eval, 1038
  - eval\_integ, 1040
  - evalf, 1038
  - evalm, 1040
  - expand, 1039
  - exponop, 1043
  - get\_point, 1041
  - get\_var, 1041
  - imag\_part, 1040
  - is\_compatible\_to, 1042
  - is\_terminating, 1042
  - is\_zero, 1042
  - ldegree, 1037
  - mul\_const, 1043
  - mul\_series, 1043
  - nops, 1036
  - normal, 1039
  - op, 1036
  - point, 1046
  - power\_const, 1044
  - precedence, 1036
  - print\_series, 1044
  - pseries, 1035, 1036
  - read\_archive, 1041
  - real\_part, 1040
  - seq, 1046
  - series, 1038
  - shift\_exponents, 1044
  - subs, 1038
  - var, 1046
- GiNaC::psi1\_SERIAL, 1047
  - serial, 1047
- GiNaC::psi2\_SERIAL, 1048
  - serial, 1048
- GiNaC::ptr< T >, 1048

- ~ptr, 1050
- get\_pointer, 1052
- makewritable, 1051
- operator!=, 1051, 1052
- operator<<, 1053
- operator->, 1051
- operator=, 1050
- operator==, 1051, 1052
- operator\*, 1050
- p, 1053
- ptr, 1050
- std::less< ptr< T > >, 1052
- swap, 1051
- GiNaC::realsymbol, 1053
  - conjugate, 1059
  - duplicate, 1059
  - get\_domain, 1059
  - imag\_part, 1059
  - real\_part, 1059
  - realsymbol, 1058, 1059
- GiNaC::refcounted, 1060
  - add\_reference, 1062
  - get\_refcount, 1062
  - refcount, 1063
  - refcounted, 1062
  - remove\_reference, 1062
  - set\_refcount, 1062
- GiNaC::registered\_class\_options, 1063
  - get\_id, 1064
  - get\_name, 1064
  - get\_parent\_name, 1064
  - get\_print\_dispatch\_table, 1064
  - name, 1065
  - parent\_name, 1065
  - print\_dispatch\_table, 1065
  - print\_func, 1064, 1065
  - registered\_class\_options, 1064
  - set\_print\_func, 1065
  - tinfo\_key, 1065
- GiNaC::relational, 1066
  - archive, 1073
  - calchash, 1075
  - canonical, 1074
  - do\_print, 1075
  - do\_print\_python\_repr, 1075
  - equal, 1071
  - eval\_ncmul, 1074
  - greater, 1072
  - greater\_or\_equal, 1072
  - info, 1072
  - less, 1071
  - less\_or\_equal, 1071
  - lh, 1077
  - lhs, 1075
  - make\_safe\_bool, 1076
  - map, 1073
  - match\_same\_type, 1074
  - nops, 1072
  - not\_equal, 1071
  - o, 1077
  - op, 1073
  - operator safe\_bool, 1076
  - operator!, 1076
  - operators, 1071
  - precedence, 1072
  - read\_archive, 1073
  - relational, 1072
  - return\_type, 1074
  - return\_type\_tinfo, 1075
  - rh, 1077
  - rhs, 1076
  - safe\_bool, 1071
  - subs, 1073
- GiNaC::relational::safe\_bool\_helper, 1089
- nonnull, 1089
- GiNaC::remember\_strategies, 1077
  - delete\_cyclic, 1078
  - delete\_lfu, 1078
  - delete\_lru, 1078
  - delete\_never, 1077
- GiNaC::remember\_table, 1078
  - add\_entry, 1080
  - clear\_all\_entries, 1080
  - init\_table, 1080
  - lookup\_entry, 1080
  - max\_assoc\_size, 1081
  - remember\_strategy, 1081
  - remember\_table, 1079
  - remember\_tables, 1080
  - show\_statistics, 1080
  - table\_size, 1081
- GiNaC::remember\_table\_entry, 1082
  - access\_counter, 1084
  - get\_last\_access, 1083
  - get\_result, 1083
  - get\_successful\_hits, 1083
  - hashvalue, 1084
  - is\_equal, 1083
  - last\_access, 1084
  - remember\_table\_entry, 1083
  - result, 1084
  - seq, 1084
  - successful\_hits, 1084
- GiNaC::remember\_table\_list, 1085
  - add\_entry, 1086
  - lookup\_entry, 1086
  - max\_assoc\_size, 1086
  - remember\_strategy, 1086
  - remember\_table\_list, 1086
- GiNaC::return\_type\_t, 1087
  - operator!=, 1087
  - operator<, 1087
  - operator==, 1087
  - rl, 1088
  - tinfo, 1088
- GiNaC::return\_types, 1088

- commutative, 1088
- noncommutative, 1089
- noncommutative\_composite, 1089
- GiNaC::scalar\_products, 1089
  - add, 1090
  - add\_vectors, 1090
  - clear, 1090
  - debugprint, 1091
  - evaluate, 1091
  - is\_defined, 1091
  - spm, 1091
- GiNaC::series\_options, 1092
  - suppress\_branchcut, 1092
- GiNaC::solve\_algo, 1092
  - automatic, 1092
  - bareiss, 1093
  - divfree, 1093
  - gauss, 1093
  - markowitz, 1093
- GiNaC::spinidx, 1093
  - archive, 1101
  - conjugate, 1101
  - do\_print, 1103
  - do\_print\_latex, 1103
  - do\_print\_tree, 1103
  - dotted, 1104
  - is\_dotted, 1102
  - is\_dummy\_pair\_same\_type, 1101
  - is\_undotted, 1102
  - match\_same\_type, 1102
  - read\_archive, 1101
  - spinidx, 1100
  - toggle\_dot, 1103
  - toggle\_variance\_dot, 1103
- GiNaC::spinmetric, 1104
  - contract\_with, 1109
  - do\_print, 1110
  - do\_print\_latex, 1110
  - eval\_indexed, 1109
  - info, 1109
- GiNaC::spmapkey, 1110
  - debugprint, 1112
  - dim, 1113
  - operator<, 1112
  - operator==, 1112
  - spmapkey, 1112
  - v1, 1112
  - v2, 1112
- GiNaC::status\_flags, 1113
  - dynamallocated, 1113
  - evaluated, 1114
  - expanded, 1114
  - has\_indices, 1114
  - has\_no\_indices, 1114
  - hash\_calculated, 1114
  - is\_negative, 1114
  - is\_positive, 1114
  - not\_shareable, 1114
  - purely\_indefinite, 1114
- GiNaC::structure< T, ComparisonPolicy >, 1114
  - add\_indexed, 1128
  - calchash, 1131
  - coeff, 1124
  - collect, 1125
  - contract\_with, 1130
  - degree, 1124
  - derivative, 1125
  - eval, 1119
  - eval\_indexed, 1120
  - eval\_ncmul, 1119
  - evalm, 1119
  - expand, 1124
  - get\_class\_name, 1119
  - get\_free\_indices, 1128
  - get\_struct, 1132
  - has, 1122
  - info, 1120
  - integer\_content, 1127
  - is\_equal\_same\_type, 1131
  - ldegree, 1124
  - let\_op, 1121
  - map, 1123
  - match, 1122
  - match\_same\_type, 1123
  - max\_coefficient, 1127
  - nops, 1121
  - normal, 1126
  - obj, 1132
  - op, 1121
  - operator->, 1131
  - operator[], 1121, 1122
  - precedence, 1120
  - print, 1120
  - return\_type, 1130
  - return\_type\_tinfo, 1131
  - scalar\_mul\_indexed, 1128
  - series, 1125
  - smod, 1127
  - structure, 1119
  - subs, 1123
  - to\_polynomial, 1126
  - to\_rational, 1126
- GiNaC::su3d, 1133
  - contract\_with, 1137
  - do\_print, 1138
  - do\_print\_latex, 1138
  - eval\_indexed, 1137
  - return\_type, 1137
- GiNaC::su3f, 1138
  - contract\_with, 1143
  - do\_print, 1143
  - do\_print\_latex, 1143
  - eval\_indexed, 1143
  - return\_type, 1143
- GiNaC::su3one, 1144
  - do\_print, 1148

- do\_print\_latex, 1148
- GiNaC::su3t, 1149
  - contract\_with, 1153
  - do\_print, 1153
  - do\_print\_latex, 1153
- GiNaC::subs\_options, 1154
  - algebraic, 1154
  - no\_index\_renaming, 1154
  - no\_pattern, 1154
  - pattern\_is\_not\_product, 1154
  - pattern\_is\_product, 1154
  - really\_subs\_idx, 1154
  - subs\_algebraic, 1154
  - subs\_no\_pattern, 1154
- GiNaC::sy\_is\_less, 1155
  - operator(), 1155
  - sy\_is\_less, 1155
  - v, 1155
- GiNaC::sy\_swap, 1155
  - operator(), 1156
  - swapped, 1156
  - sy\_swap, 1156
  - v, 1156
- GiNaC::sym\_desc, 1157
  - deg\_a, 1159
  - deg\_b, 1159
  - lddeg\_a, 1159
  - lddeg\_b, 1159
  - max\_deg, 1159
  - max\_lcnops, 1159
  - operator<, 1158
  - sym, 1159
  - sym\_desc, 1158
- GiNaC::symbol, 1160
  - archive, 1168
  - calchash, 1169
  - conjugate, 1167
  - derivative, 1168
  - do\_print, 1170
  - do\_print\_latex, 1170
  - do\_print\_python\_repr, 1170
  - do\_print\_tree, 1170
  - eval, 1165
  - evalf, 1165
  - get\_domain, 1169
  - get\_name, 1170
  - get\_TeX\_name, 1170
  - imag\_part, 1167
  - info, 1165
  - is\_equal\_same\_type, 1168
  - is\_polynomial, 1167
  - name, 1171
  - next\_serial, 1171
  - normal, 1166
  - read\_archive, 1168
  - real\_part, 1167
  - serial, 1171
  - series, 1166
  - set\_name, 1169
  - set\_TeX\_name, 1169
  - subs, 1166
  - symbol, 1165
  - TeX\_name, 1171
  - to\_polynomial, 1167
  - to\_rational, 1166
- GiNaC::symbolset, 1172
  - has, 1172
  - insert\_symbols, 1172
  - s, 1173
  - symbolset, 1172
- GiNaC::symmetry, 1173
  - add, 1180
  - antisymmetric, 1178
  - archive, 1179
  - calchash, 1179
  - canonicalize, 1182
  - children, 1183
  - cyclic, 1178
  - do\_print, 1181
  - do\_print\_tree, 1181
  - get\_type, 1180
  - has\_cyclic, 1181
  - has\_nonsymmetric, 1181
  - has\_symmetry, 1181
  - indices, 1183
  - none, 1178
  - read\_archive, 1179
  - set\_type, 1180
  - sy\_is\_less, 1182
  - sy\_swap, 1182
  - symmetric, 1178
  - symmetry, 1178
  - symmetry\_type, 1178
  - type, 1183
  - validate, 1180
- GiNaC::symminfo, 1183
  - coeff, 1185
  - num, 1185
  - orig, 1185
  - symminfo, 1185
  - symmterm, 1185
- GiNaC::symminfo\_is\_less\_by\_orig, 1186
  - operator(), 1186
- GiNaC::symminfo\_is\_less\_by\_symmterm, 1186
  - operator(), 1186
- GiNaC::tensdelta, 1187
  - contract\_with, 1192
  - do\_print, 1192
  - do\_print\_latex, 1192
  - eval\_indexed, 1191
  - info, 1191
  - return\_type, 1192
- GiNaC::tensepsilon, 1193
  - archive, 1198
  - contract\_with, 1198
  - do\_print, 1199

- do\_print\_latex, 1199
- eval\_indexed, 1198
- info, 1198
- minkowski, 1199
- pos\_sig, 1199
- read\_archive, 1198
- return\_type, 1199
- tensepsilon, 1197
- GiNaC::tensmetric, 1200
  - contract\_with, 1205
  - do\_print, 1205
  - eval\_indexed, 1205
  - info, 1205
  - return\_type, 1205
- GiNaC::tensor, 1206
  - replace\_contr\_index, 1210
  - return\_type, 1210
- GiNaC::terminfo, 1211
  - orig, 1212
  - symm, 1212
  - terminfo, 1212
- GiNaC::terminfo\_is\_less, 1213
  - operator(), 1213
- GiNaC::unarchive\_table\_t, 1218
  - ~unarchive\_table\_t, 1218
  - find, 1218
  - insert, 1218
  - unarch\_map, 1219
  - unarchive\_table\_t, 1218
  - usecount, 1219
- GiNaC::user\_defined\_kernel, 1219
  - do\_print, 1226
  - f, 1226
  - is\_numeric, 1225
  - Laurent\_series, 1225
  - let\_op, 1225
  - nops, 1225
  - op, 1225
  - user\_defined\_kernel, 1225
  - uses\_Laurent\_series, 1226
  - x, 1226
- GiNaC::varidx, 1227
  - archive, 1233
  - covariant, 1235
  - do\_print, 1235
  - do\_print\_tree, 1235
  - is\_contravariant, 1234
  - is\_covariant, 1234
  - is\_dummy\_pair\_same\_type, 1233
  - match\_same\_type, 1234
  - read\_archive, 1233
  - toggle\_variance, 1235
  - varidx, 1233
- GiNaC::visitor, 1236
  - ~visitor, 1236
- GiNaC::wildcard, 1236
  - archive, 1241
  - calchash, 1241
  - do\_print, 1242
  - do\_print\_python\_repr, 1242
  - do\_print\_tree, 1242
  - get\_label, 1241
  - label, 1242
  - match, 1241
  - read\_archive, 1241
  - wildcard, 1240
- GiNaC::zeta1\_SERIAL, 1243
  - serial, 1243
- GiNaC::zeta2\_SERIAL, 1243
  - serial, 1244
- GINAC\_ASSERT
  - assertion.h, 1250
- GINAC\_BIND\_UNARCHIVER
  - archive.h, 1249
  - GiNaC, 92, 97, 98, 110, 111, 116, 131, 132, 136, 199, 201–204, 206, 211, 212, 228, 260, 261, 263–265, 272, 281, 284
- GINAC\_DECLARE\_PRINT\_CONTEXT
  - print.h, 1340
- GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE
  - print.h, 1340
- GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON
  - print.h, 1340
- GINAC\_DECLARE\_REGISTERED\_CLASS
  - registrar.h, 1345
- GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON
  - registrar.h, 1344
- GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS
  - registrar.h, 1344
- GINAC\_DECLARE\_UNARCHIVER
  - archive.h, 1248
  - GiNaC, 93, 109, 115, 116, 130–132, 134, 141, 199, 204–206, 208, 212, 245, 260, 261, 263, 264, 268, 276, 282
- GINAC\_IMPLEMENT\_PRINT\_CONTEXT
  - print.h, 1340
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS
  - registrar.h, 1345
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT
  - GiNaC, 92, 94, 97, 110, 115, 129–132, 136, 198, 201–204, 206, 211, 212, 227, 259, 261, 263, 264, 271, 281
  - registrar.h, 1345
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T
  - registrar.h, 1346
- GINAC\_LT\_AGE
  - version.h, 1362
- GINAC\_LT\_CURRENT
  - version.h, 1362
- GINAC\_LT\_REVISION
  - version.h, 1362
- GINACLIB\_ARCHIVE\_AGE
  - version.h, 1363
- GINACLIB\_ARCHIVE\_VERSION
  - version.h, 1363
- GINACLIB\_MAJOR\_VERSION

- version.h, [1362](#)
- GINACLIB\_MICRO\_VERSION
  - version.h, [1362](#)
- GINACLIB\_MINOR\_VERSION
  - version.h, [1362](#)
- GINACLIB\_STR
  - version.h, [1363](#)
- GINACLIB\_STR\_HELPER
  - version.h, [1363](#)
- GINACLIB\_VERSION
  - version.h, [1363](#)
- golden\_ratio\_hash
  - GiNaC, [277](#)
- greater
  - GiNaC::relational, [1072](#)
- greater\_or\_equal
  - GiNaC::relational, [1072](#)
- Gt
  - GiNaC::Gt, [699](#)
  - GiNaC::Gt::kernel, [772](#)
- Gt.cpp, [1291](#)
  - DEBUG\_GT, [1292](#)
- Gt.h, [1292](#)
- Gt\_helpers.cpp, [1292](#)
- Gt\_helpers.h, [1293](#)
- guess\_precision
  - GiNaC, [237](#)
- H\_deriv
  - GiNaC, [175](#)
- H\_eval
  - GiNaC, [175](#)
- H\_evalf
  - GiNaC, [175](#)
- H\_print\_latex
  - GiNaC, [175](#)
- H\_series
  - GiNaC, [175](#)
- handle\_factor
  - GiNaC::make\_flat\_inserter, [796](#)
- has
  - GiNaC, [118](#)
  - GiNaC::basic, [356](#)
  - GiNaC::ex, [562](#)
  - GiNaC::Gt, [702](#)
  - GiNaC::mul, [849](#)
  - GiNaC::numeric, [922](#)
  - GiNaC::power, [991](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1122](#)
  - GiNaC::symbolset, [1172](#)
- has\_cyclic
  - GiNaC::symmetry, [1181](#)
- has\_derivative
  - GiNaC::function\_options, [685](#)
- has\_dummy\_index\_for
  - GiNaC::indexed, [743](#)
- has\_ex
  - GiNaC::archive\_node, [341](#)
- has\_expression
  - GiNaC::archive\_node, [343](#)
- has\_indices
  - GiNaC::info\_flags, [747](#)
  - GiNaC::status\_flags, [1114](#)
- has\_no\_indices
  - GiNaC::status\_flags, [1114](#)
- has\_nonsymmetric
  - GiNaC::symmetry, [1181](#)
- has\_power
  - GiNaC::function\_options, [685](#)
- has\_same\_ex\_as
  - GiNaC::archive\_node, [341](#)
- has\_symmetry
  - GiNaC::symmetry, [1181](#)
- has\_trailing\_zero
  - GiNaC::integration\_kernel, [764](#)
- hash\_calculated
  - GiNaC::status\_flags, [1114](#)
- hash\_map.h, [1294](#)
- hash\_seed.h, [1294](#)
- hashvalue
  - GiNaC::basic, [375](#)
  - GiNaC::remember\_table\_entry, [1084](#)
- hasindex
  - GiNaC, [138](#)
- haswild
  - GiNaC, [282](#)
- head
  - GiNaC::composition\_generator::coolmulti, [474](#)
- heur\_gcd
  - GiNaC, [220](#)
- heur\_gcd\_z
  - GiNaC, [220](#)
- hold
  - GiNaC::basic, [371](#)
- hold\_ncmul
  - GiNaC, [212](#)
  - GiNaC::ncmul, [910](#)
- I
  - GiNaC, [285](#)
- i
  - GiNaC::composition\_generator::coolmulti, [474](#)
  - GiNaC::const\_iterator, [436](#)
  - GiNaC::internal::\_iter\_rep, [306](#)
- i\_end
  - GiNaC::internal::\_iter\_rep, [307](#)
- id
  - GiNaC::print\_context\_options, [1002](#)
- identify\_parents
  - GiNaC::class\_info< OPT >, [393](#)
- idx
  - GiNaC, [284](#)
  - GiNaC::idx, [719](#)
  - GiNaC::info\_flags, [747](#)
- idx.cpp, [1295](#)
- idx.h, [1296](#)
- idx\_symmetrization
  - GiNaC, [138](#)

- ifactor
  - GiNaC, [199](#)
- imag
  - GiNaC, [249](#)
  - GiNaC::numeric, [937](#)
- imag\_part
  - GiNaC, [118](#)
  - GiNaC::add, [322](#)
  - GiNaC::basic, [366](#)
  - GiNaC::constant, [450](#)
  - GiNaC::container< C >, [464](#)
  - GiNaC::ex, [562](#)
  - GiNaC::function, [644](#)
  - GiNaC::indexed, [740](#)
  - GiNaC::matrix, [809](#)
  - GiNaC::mul, [850](#)
  - GiNaC::ncmul, [908](#)
  - GiNaC::numeric, [925](#)
  - GiNaC::power, [993](#)
  - GiNaC::pseries, [1040](#)
  - GiNaC::realsymbol, [1059](#)
  - GiNaC::symbol, [1167](#)
- imag\_part\_conjugate
  - GiNaC, [145](#)
- imag\_part\_eval
  - GiNaC, [144](#)
- imag\_part\_evalf
  - GiNaC, [144](#)
- imag\_part\_expl\_derivative
  - GiNaC, [145](#)
- imag\_part\_f
  - GiNaC::function\_options, [688](#)
- imag\_part\_func
  - GiNaC::function\_options, [666–668](#), [681](#)
- imag\_part\_funcp
  - GiNaC, [70](#)
- imag\_part\_funcp\_1
  - GiNaC, [71](#)
- imag\_part\_funcp\_10
  - GiNaC, [83](#)
- imag\_part\_funcp\_11
  - GiNaC, [84](#)
- imag\_part\_funcp\_12
  - GiNaC, [86](#)
- imag\_part\_funcp\_13
  - GiNaC, [87](#)
- imag\_part\_funcp\_14
  - GiNaC, [89](#)
- imag\_part\_funcp\_2
  - GiNaC, [73](#)
- imag\_part\_funcp\_3
  - GiNaC, [74](#)
- imag\_part\_funcp\_4
  - GiNaC, [75](#)
- imag\_part\_funcp\_5
  - GiNaC, [76](#)
- imag\_part\_funcp\_6
  - GiNaC, [78](#)
- imag\_part\_funcp\_7
  - GiNaC, [79](#)
- imag\_part\_funcp\_8
  - GiNaC, [80](#)
- imag\_part\_funcp\_9
  - GiNaC, [82](#)
- imag\_part\_funcp\_exvector
  - GiNaC, [90](#)
- imag\_part\_imag\_part
  - GiNaC, [145](#)
- imag\_part\_print\_latex
  - GiNaC, [145](#)
- imag\_part\_real\_part
  - GiNaC, [145](#)
- imag\_part\_use\_exvector\_args
  - GiNaC::function\_options, [691](#)
- impl
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, [1215](#)
  - GiNaC::print\_functor, [1013](#)
- increment
  - GiNaC::const\_postorder\_iterator, [439](#)
  - GiNaC::const\_preorder\_iterator, [442](#)
- indefinite
  - GiNaC::info\_flags, [747](#)
- index0
  - GiNaC, [265](#)
- index1
  - GiNaC, [265](#)
- index2
  - GiNaC, [265](#)
- index3
  - GiNaC, [265](#)
- index\_dimensions
  - GiNaC, [259](#)
- indexed
  - GiNaC::indexed, [734–736](#), [738](#), [739](#)
  - GiNaC::info\_flags, [747](#)
- indexed.cpp, [1297](#)
- indexed.h, [1299](#)
- indices
  - GiNaC::symmetry, [1183](#)
- indices\_consistent
  - GiNaC, [136](#)
- info
  - GiNaC::add, [317](#)
  - GiNaC::basic, [354](#)
  - GiNaC::class\_info< OPT >::tree\_node, [1217](#)
  - GiNaC::constant, [449](#)
  - GiNaC::container< C >, [461](#)
  - GiNaC::ex, [558](#)
  - GiNaC::expairseq, [596](#)
  - GiNaC::function, [645](#)
  - GiNaC::idx, [720](#)
  - GiNaC::indexed, [739](#)
  - GiNaC::minkmetric, [828](#)
  - GiNaC::mul, [847](#)
  - GiNaC::ncmul, [905](#)

- GiNaC::numeric, [920](#)
  - GiNaC::power, [987](#)
  - GiNaC::relational, [1072](#)
  - GiNaC::spinmetric, [1109](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1120](#)
  - GiNaC::symbol, [1165](#)
  - GiNaC::tensdelta, [1191](#)
  - GiNaC::tensepsilon, [1198](#)
  - GiNaC::tensmetric, [1205](#)
- info\_f
  - GiNaC::function\_options, [689](#)
- info\_func
  - GiNaC::function\_options, [678–681](#)
- info\_funcp
  - GiNaC, [71](#)
- info\_funcp\_1
  - GiNaC, [72](#)
- info\_funcp\_10
  - GiNaC, [84](#)
- info\_funcp\_11
  - GiNaC, [85](#)
- info\_funcp\_12
  - GiNaC, [87](#)
- info\_funcp\_13
  - GiNaC, [88](#)
- info\_funcp\_14
  - GiNaC, [90](#)
- info\_funcp\_2
  - GiNaC, [73](#)
- info\_funcp\_3
  - GiNaC, [74](#)
- info\_funcp\_4
  - GiNaC, [76](#)
- info\_funcp\_5
  - GiNaC, [77](#)
- info\_funcp\_6
  - GiNaC, [78](#)
- info\_funcp\_7
  - GiNaC, [80](#)
- info\_funcp\_8
  - GiNaC, [81](#)
- info\_funcp\_9
  - GiNaC, [82](#)
- info\_funcp\_exvector
  - GiNaC, [91](#)
- info\_use\_exvector\_args
  - GiNaC::function\_options, [691](#)
- inifcns.cpp, [1300](#)
- inifcns.h, [1303](#)
- inifcns\_elliptic.cpp, [1304](#)
- inifcns\_gamma.cpp, [1306](#)
- inifcns\_nstdsums.cpp, [1307](#)
- inifcns\_trans.cpp, [1309](#)
- init
  - GiNaC::basic\_multi\_iterator< T >, [386](#)
  - GiNaC::multi\_iterator\_counter< T >, [861](#)
  - GiNaC::multi\_iterator\_counter\_indv< T >, [865](#)
  - GiNaC::multi\_iterator\_ordered< T >, [869](#)
  - GiNaC::multi\_iterator\_ordered\_eq< T >, [873](#)
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, [876](#)
  - GiNaC::multi\_iterator\_permutation< T >, [881](#)
  - GiNaC::multi\_iterator\_shuffle< T >, [885](#)
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, [889](#)
- init\_table
  - GiNaC::remember\_table, [1080](#)
- init\_unarchivers
  - GiNaC::library\_init, [794](#)
- initialize
  - GiNaC::function\_options, [657](#)
- insert
  - GiNaC::unarchive\_table\_t, [1218](#)
- insert\_symbols
  - GiNaC::symbolset, [1172](#)
- int\_length
  - GiNaC::numeric, [938](#)
- integer
  - GiNaC::info\_flags, [747](#)
- integer\_content
  - GiNaC::add, [320](#)
  - GiNaC::basic, [362](#)
  - GiNaC::ex, [571](#)
  - GiNaC::mul, [851](#)
  - GiNaC::numeric, [924](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1127](#)
- integer\_partition
  - GiNaC::Gt\_detail, [301](#)
- integer\_polynomial
  - GiNaC::info\_flags, [747](#)
- integral
  - GiNaC::error\_and\_integral, [545](#)
  - GiNaC::integral, [753](#)
- integral.cpp, [1312](#)
- integral.h, [1313](#)
- integrate
  - GiNaC::Gt\_detail::pathintegral\_term, [948](#)
- integration\_kernel.cpp, [1313](#)
- integration\_kernel.h, [1315](#)
- interpolate
  - GiNaC, [219](#)
- inv\_at\_cit
  - GiNaC::archive, [328](#)
- inverse
  - GiNaC, [210](#), [245](#)
  - GiNaC::matrix, [816](#)
  - GiNaC::numeric, [930](#)
- inverse\_atoms
  - GiNaC::archive, [333](#)
- iquo
  - GiNaC, [242](#), [243](#)
- irem
  - GiNaC, [241](#), [242](#)
- is\_a
  - GiNaC, [94](#), [125](#), [261](#)
  - GiNaC::ex, [581](#)
- is\_canonical
  - GiNaC::expairseq, [606](#)



- is\_canonical\_numeric
  - GiNaC::expair, [587](#)
- is\_cinteger
  - GiNaC, [248](#)
  - GiNaC::numeric, [934](#)
- is\_clifford\_tinfo
  - GiNaC, [109](#)
- is\_color\_tinfo
  - GiNaC, [113](#)
- is\_compatible\_to
  - GiNaC::pseries, [1042](#)
- is\_contravariant
  - GiNaC::varidx, [1234](#)
- is\_covariant
  - GiNaC::varidx, [1234](#)
- is\_crational
  - GiNaC, [248](#)
  - GiNaC::numeric, [934](#)
- is\_defined
  - GiNaC::scalar\_products, [1091](#)
- is\_dim\_numeric
  - GiNaC::idx, [724](#)
- is\_dim\_symbolic
  - GiNaC::idx, [724](#)
- is\_dirac\_slash
  - GiNaC, [98](#)
- is\_discriminant\_of\_quadratic\_number\_field
  - GiNaC, [199](#)
- is\_dotted
  - GiNaC::spinidx, [1102](#)
- is\_dummy\_pair
  - GiNaC, [133](#)
- is\_dummy\_pair\_same\_type
  - GiNaC::idx, [723](#)
  - GiNaC::spinidx, [1101](#)
  - GiNaC::varidx, [1233](#)
- is\_equal
  - GiNaC::basic, [370](#)
  - GiNaC::ex, [574](#)
  - GiNaC::expair, [586](#)
  - GiNaC::numeric, [931](#)
  - GiNaC::remember\_table\_entry, [1083](#)
- is\_equal\_same\_type
  - GiNaC::basic, [367](#)
  - GiNaC::constant, [451](#)
  - GiNaC::container< C >, [464](#)
  - GiNaC::expairseq, [599](#)
  - GiNaC::fderivative, [627](#)
  - GiNaC::function, [646](#)
  - GiNaC::numeric, [926](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1131](#)
  - GiNaC::symbol, [1168](#)
- is\_even
  - GiNaC, [247](#)
  - GiNaC::numeric, [933](#)
- is\_ex\_the\_function
  - function.h, [1289](#)
- is\_exactly\_a
  - GiNaC, [95](#), [125](#)
  - GiNaC::ex, [581](#)
- is\_integer
  - GiNaC, [246](#)
  - GiNaC::numeric, [932](#)
- is\_less
  - GiNaC::expair, [586](#)
- is\_negative
  - GiNaC, [246](#)
  - GiNaC::numeric, [932](#)
  - GiNaC::status\_flags, [1114](#)
- is\_nonneg\_integer
  - GiNaC, [247](#)
  - GiNaC::numeric, [933](#)
- is\_numeric
  - GiNaC::Ebar\_kernel, [511](#)
  - GiNaC::Eisenstein\_h\_kernel, [521](#)
  - GiNaC::Eisenstein\_kernel, [531](#)
  - GiNaC::ELi\_kernel, [541](#)
  - GiNaC::idx, [723](#)
  - GiNaC::integration\_kernel, [764](#)
  - GiNaC::Kronecker\_dtau\_kernel, [779](#)
  - GiNaC::Kronecker\_dz\_kernel, [788](#)
  - GiNaC::modular\_form\_kernel, [837](#)
  - GiNaC::multiple\_polylog\_kernel, [896](#)
  - GiNaC::user\_defined\_kernel, [1225](#)
- is\_odd
  - GiNaC, [247](#)
  - GiNaC::numeric, [933](#)
- is\_order\_function
  - GiNaC, [162](#)
- is\_polynomial
  - GiNaC, [118](#)
  - GiNaC::add, [318](#)
  - GiNaC::basic, [358](#)
  - GiNaC::constant, [449](#)
  - GiNaC::ex, [565](#)
  - GiNaC::mul, [848](#)
  - GiNaC::numeric, [921](#)
  - GiNaC::power, [988](#)
  - GiNaC::symbol, [1167](#)
- is\_pos\_integer
  - GiNaC, [247](#)
  - GiNaC::numeric, [932](#)
- is\_positive
  - GiNaC, [246](#)
  - GiNaC::numeric, [932](#)
  - GiNaC::status\_flags, [1114](#)
- is\_prime
  - GiNaC, [247](#)
  - GiNaC::numeric, [933](#)
- is\_rational
  - GiNaC, [247](#)
  - GiNaC::numeric, [933](#)
- is\_real
  - GiNaC, [248](#)
  - GiNaC::numeric, [934](#)
- is\_symbolic

- GiNaC::idx, [723](#)
- is\_terminating
  - GiNaC, [262](#)
  - GiNaC::pseries, [1042](#)
- is\_the\_function
  - GiNaC, [131](#)
- is\_the\_function< G\_SERIAL >
  - GiNaC, [161](#)
- is\_the\_function< iterated\_integral\_SERIAL >
  - GiNaC, [162](#)
- is\_the\_function< psi\_SERIAL >
  - GiNaC, [161](#)
- is\_the\_function< zeta\_SERIAL >
  - GiNaC, [160](#)
- is\_undotted
  - GiNaC::spinidx, [1102](#)
- is\_valid
  - GiNaC::print\_functor, [1013](#)
- is\_zero
  - GiNaC, [124](#), [246](#)
  - GiNaC::ex, [575](#)
  - GiNaC::numeric, [931](#)
  - GiNaC::pseries, [1042](#)
- is\_zero\_matrix
  - GiNaC::ex, [575](#)
  - GiNaC::matrix, [818](#)
- isqrt
  - GiNaC, [244](#)
- iterated\_integral
  - GiNaC, [161](#), [162](#)
- iterated\_integral2\_eval
  - GiNaC, [165](#)
- iterated\_integral2\_evalf
  - GiNaC, [165](#)
- iterated\_integral3\_eval
  - GiNaC, [166](#)
- iterated\_integral3\_evalf
  - GiNaC, [165](#)
- iterated\_integral\_evalf\_impl
  - GiNaC, [165](#)
- iterator\_category
  - GiNaC::const\_iterator, [432](#)
  - GiNaC::const\_postorder\_iterator, [437](#)
  - GiNaC::const\_preorder\_iterator, [440](#)
- K
  - GiNaC::Eisenstein\_kernel, [533](#)
  - GiNaC::Kronecker\_dtau\_kernel, [781](#)
  - GiNaC::Kronecker\_dz\_kernel, [790](#)
- k
  - GiNaC::Eisenstein\_h\_kernel, [523](#)
  - GiNaC::Eisenstein\_kernel, [532](#)
  - GiNaC::modular\_form\_kernel, [838](#)
- kernel
  - GiNaC::Gt::kernel, [772](#)
- Kronecker\_dtau\_kernel
  - GiNaC::Kronecker\_dtau\_kernel, [779](#)
- Kronecker\_dz\_kernel
  - GiNaC::Kronecker\_dz\_kernel, [788](#)
- kronecker\_symbol
  - GiNaC, [200](#)
- label
  - GiNaC::wildcard, [1242](#)
- lanczos\_coeffs
  - GiNaC::lanczos\_coeffs, [791](#)
- laplace
  - GiNaC::determinant\_algo, [477](#)
- last\_access
  - GiNaC::remember\_table\_entry, [1084](#)
- latex
  - GiNaC, [258](#)
- latex\_name
  - GiNaC::function\_options, [658](#)
- Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, [521](#)
  - GiNaC::Eisenstein\_kernel, [531](#)
  - GiNaC::integration\_kernel, [765](#)
  - GiNaC::modular\_form\_kernel, [837](#)
  - GiNaC::user\_defined\_kernel, [1225](#)
- lcm
  - GiNaC, [223](#), [243](#)
- lcm\_of\_coefficients\_denominators
  - GiNaC, [215](#)
- lcmcoeff
  - GiNaC, [214](#)
- lcoeff
  - GiNaC::ex, [565](#)
- ldeg\_a
  - GiNaC::sym\_desc, [1159](#)
- ldeg\_b
  - GiNaC::sym\_desc, [1159](#)
- ldegree
  - GiNaC, [119](#)
  - GiNaC::add, [318](#)
  - GiNaC::basic, [359](#)
  - GiNaC::ex, [565](#)
  - GiNaC::integral, [754](#)
  - GiNaC::mul, [848](#)
  - GiNaC::ncmul, [905](#)
  - GiNaC::numeric, [921](#)
  - GiNaC::power, [989](#)
  - GiNaC::pseries, [1037](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1124](#)
- less
  - GiNaC::relational, [1071](#)
- less\_or\_equal
  - GiNaC::relational, [1071](#)
- let\_op
  - GiNaC::basic, [356](#)
  - GiNaC::clifford, [406](#)
  - GiNaC::container< C >, [462](#)
  - GiNaC::Ebar\_kernel, [511](#)
  - GiNaC::Eisenstein\_h\_kernel, [520](#)
  - GiNaC::Eisenstein\_kernel, [531](#)
  - GiNaC::ELi\_kernel, [541](#)
  - GiNaC::ex, [560](#)
  - GiNaC::integral, [755](#)

- GiNaC::Kronecker\_dtau\_kernel, 779
  - GiNaC::Kronecker\_dz\_kernel, 788
  - GiNaC::matrix, 807
  - GiNaC::modular\_form\_kernel, 836
  - GiNaC::multiple\_polylog\_kernel, 896
  - GiNaC::structure< T, ComparisonPolicy >, 1121
  - GiNaC::user\_defined\_kernel, 1225
- lgamma
  - GiNaC, 237
- lgamma\_conjugate
  - GiNaC, 167
- lgamma\_deriv
  - GiNaC, 166
- lgamma\_eval
  - GiNaC, 166
- lgamma\_evalf
  - GiNaC, 166
- lgamma\_series
  - GiNaC, 167
- lh
  - GiNaC::relational, 1077
- lhs
  - GiNaC, 123
  - GiNaC::ex, 561
  - GiNaC::relational, 1075
- Li2
  - GiNaC, 236
- Li2\_
  - GiNaC, 236
- Li2\_conjugate
  - GiNaC, 153
- Li2\_deriv
  - GiNaC, 153
- Li2\_eval
  - GiNaC, 153
- Li2\_evalf
  - GiNaC, 152
- Li2\_projection
  - GiNaC, 236
- Li2\_series
  - GiNaC, 153, 235
- Li3\_eval
  - GiNaC, 154
- Li\_deriv
  - GiNaC, 173
- Li\_eval
  - GiNaC, 172
- Li\_evalf
  - GiNaC, 172
- Li\_print\_latex
  - GiNaC, 173
- Li\_series
  - GiNaC, 172
- library\_init
  - GiNaC::library\_init, 794
- library\_initializer
  - GiNaC, 284
- likely
  - compiler.h, 1262
- link\_ex
  - GiNaC, 128, 129
- list
  - GiNaC::info\_flags, 747
- log
  - GiNaC, 230
- log2
  - GiNaC, 276
- log\_conjugate
  - GiNaC, 181
- log\_deriv
  - GiNaC, 180
- log\_eval
  - GiNaC, 179
- log\_evalf
  - GiNaC, 179
- log\_expand
  - GiNaC, 180
- log\_imag\_part
  - GiNaC, 180
- log\_info
  - GiNaC, 181
- log\_real\_part
  - GiNaC, 180
- log\_series
  - GiNaC, 180
- lookup\_entry
  - GiNaC::remember\_table, 1080
  - GiNaC::remember\_table\_list, 1086
- lookup\_map
  - GiNaC, 91
- lookup\_remember\_table
  - GiNaC::function, 647
- lorentz\_eps
  - GiNaC, 275
- lorentz\_g
  - GiNaC, 273
- lsolve
  - GiNaC, 159
- lst
  - GiNaC, 91
- lst.cpp, 1316
- lst.h, 1317
- lst\_evaluate
  - GiNaC::Gt, 706, 707
- lst\_to\_clifford
  - GiNaC, 105
- lst\_to\_matrix
  - GiNaC, 206
- m
  - GiNaC::basic\_partition\_generator::mpartition2, 840
  - GiNaC::Ebar\_kernel, 513
  - GiNaC::ELi\_kernel, 542
  - GiNaC::matrix, 822
  - GiNaC::partition\_with\_zero\_parts\_generator, 945
- make\_flat

- GiNaC::expairseq, 605
- make\_flat\_inserter
  - GiNaC::make\_flat\_inserter, 796
- make\_real\_float
  - GiNaC, 227
- make\_return\_type\_t
  - GiNaC, 262
- make\_safe\_bool
  - GiNaC::relational, 1076
- makewritable
  - GiNaC::ptr< T >, 1051
- makewriteable
  - GiNaC::ex, 579
- map
  - GiNaC::basic, 358
  - GiNaC::ex, 563, 564
  - GiNaC::expairseq, 597
  - GiNaC::idx, 720
  - GiNaC::power, 988
  - GiNaC::relational, 1073
  - GiNaC::structure< T, ComparisonPolicy >, 1123
- map\_eval\_integ
  - GiNaC, 282
- map\_evalm
  - GiNaC, 282
- markowitz
  - GiNaC::solve\_algo, 1093
- markowitz\_elimination
  - GiNaC::matrix, 820
- match
  - GiNaC, 121
  - GiNaC::basic, 357
  - GiNaC::ex, 562
  - GiNaC::expairseq, 598
  - GiNaC::Gt, 702
  - GiNaC::structure< T, ComparisonPolicy >, 1122
  - GiNaC::wildcard, 1241
- match\_same\_type
  - GiNaC::basic, 357
  - GiNaC::clifford, 403
  - GiNaC::color, 423
  - GiNaC::fderivative, 627
  - GiNaC::function, 646
  - GiNaC::idx, 722
  - GiNaC::matrix, 810
  - GiNaC::relational, 1074
  - GiNaC::spinidx, 1102
  - GiNaC::structure< T, ComparisonPolicy >, 1123
  - GiNaC::varidx, 1234
- matrix
  - GiNaC::matrix, 805, 806
- matrix.cpp, 1317
- matrix.h, 1318
- max\_assoc\_size
  - GiNaC::remember\_table, 1081
  - GiNaC::remember\_table\_list, 1086
- max\_coefficient
  - GiNaC::add, 321
- GiNaC::basic, 363
  - GiNaC::ex, 573
  - GiNaC::mul, 851
  - GiNaC::numeric, 924
  - GiNaC::structure< T, ComparisonPolicy >, 1127
- max\_deg
  - GiNaC::sym\_desc, 1159
- max\_integration\_level
  - GiNaC::integral, 758
- max\_1cnops
  - GiNaC::sym\_desc, 1159
- metric
  - GiNaC::clifford, 407
- metric\_tensor
  - GiNaC, 273
- minimal\_dim
  - GiNaC, 133
  - GiNaC::idx, 724
- minkmetric
  - GiNaC::minkmetric, 828
- minkowski
  - GiNaC::tensepsilon, 1199
- mod
  - GiNaC, 241
- modular\_form\_kernel
  - GiNaC::modular\_form\_kernel, 836
- mpartition2
  - GiNaC::basic\_partition\_generator::mpartition2, 839
- mpgen
  - GiNaC::basic\_partition\_generator, 390
- mul
  - GiNaC::add, 326
  - GiNaC::matrix, 811
  - GiNaC::mul, 846, 847
  - GiNaC::numeric, 927
  - GiNaC::power, 998
- mul.cpp, 1320
- mul.h, 1320
- mul\_const
  - GiNaC::pseries, 1043
- mul\_dyn
  - GiNaC::numeric, 928
- mul\_scalar
  - GiNaC::matrix, 812
- mul\_series
  - GiNaC::pseries, 1043
- multi\_iterator\_counter
  - GiNaC::multi\_iterator\_counter< T >, 861
- multi\_iterator\_counter\_indv
  - GiNaC::multi\_iterator\_counter\_indv< T >, 864, 865
- multi\_iterator\_ordered
  - GiNaC::multi\_iterator\_ordered< T >, 868, 869
- multi\_iterator\_ordered\_eq
  - GiNaC::multi\_iterator\_ordered\_eq< T >, 872
- multi\_iterator\_ordered\_eq\_indv
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 876

- multi\_iterator\_permutation
  - GiNaC::multi\_iterator\_permutation< T >, 880
- multi\_iterator\_shuffle
  - GiNaC::multi\_iterator\_shuffle< T >, 884
- multi\_iterator\_shuffle\_prime
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, 889
- multinomial\_coefficient
  - GiNaC, 276
- multiple\_polylog\_kernel
  - GiNaC::multiple\_polylog\_kernel, 896
- multiply\_lcm
  - GiNaC, 215
- my\_ios\_callback
  - GiNaC, 256
- my\_ios\_index
  - GiNaC, 256
- N
  - GiNaC::basic\_multi\_iterator< T >, 387
  - GiNaC::Eisenstein\_h\_kernel, 523
  - GiNaC::Eisenstein\_kernel, 532
- n
  - GiNaC::basic\_partition\_generator::mpartition2, 840
  - GiNaC::Ebar\_kernel, 513
  - GiNaC::ELi\_kernel, 542
  - GiNaC::Kronecker\_dtau\_kernel, 781
  - GiNaC::Kronecker\_dz\_kernel, 790
- N\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, 885
- name
  - GiNaC::archive::archived\_ex, 346
  - GiNaC::archive\_node::property, 1028
  - GiNaC::archive\_node::property\_info, 1030
  - GiNaC::constant, 453
  - GiNaC::function\_options, 687
  - GiNaC::print\_context\_options, 1002
  - GiNaC::registered\_class\_options, 1065
  - GiNaC::symbol, 1171
- nargs
  - GiNaC::Gt, 709
- ncmul
  - GiNaC::mul, 858
  - GiNaC::ncmul, 904
- ncmul.cpp, 1321
- ncmul.h, 1322
- negative
  - GiNaC::info\_flags, 747
- negint
  - GiNaC::info\_flags, 747
- next
  - GiNaC::class\_info< OPT >, 393
  - GiNaC::composition\_generator, 429
  - GiNaC::composition\_generator::coolmulti::element, 534
  - GiNaC::partition\_generator, 942
  - GiNaC::partition\_with\_zero\_parts\_generator, 945
- next\_partition
  - GiNaC::basic\_partition\_generator::mpartition2, 839
- next\_permutation
  - GiNaC::composition\_generator::coolmulti, 474
- next\_print\_context\_id
  - GiNaC, 285
- next\_serial
  - GiNaC::constant, 453
  - GiNaC::symbol, 1171
- ni
  - GiNaC::Gt::kernel, 773
- no\_heur\_gcd
  - GiNaC::gcd\_options, 693
- no\_index\_dimensions
  - GiNaC, 259
- no\_index\_renaming
  - GiNaC::subs\_options, 1154
- no\_part\_factored
  - GiNaC::gcd\_options, 694
- no\_pattern
  - GiNaC::subs\_options, 1154
- no\_type
  - GiNaC::has\_distance< T >, 711
- nodes
  - GiNaC::archive, 333
- noncommutative
  - GiNaC::return\_types, 1089
- noncommutative\_composite
  - GiNaC::return\_types, 1089
- none
  - GiNaC::symmetry, 1178
- nonnegative
  - GiNaC::info\_flags, 747
- nonnegint
  - GiNaC::info\_flags, 747
- nonnull
  - GiNaC::relational::safe\_bool\_helper, 1089
- nops
  - GiNaC, 117, 208
  - GiNaC::basic, 355
  - GiNaC::clifford, 405
  - GiNaC::container< C >, 461
  - GiNaC::Ebar\_kernel, 511
  - GiNaC::Eisenstein\_h\_kernel, 520
  - GiNaC::Eisenstein\_kernel, 530
  - GiNaC::ELi\_kernel, 541
  - GiNaC::ex, 559
  - GiNaC::expairseq, 596
  - GiNaC::Gt, 701
  - GiNaC::idx, 720
  - GiNaC::integral, 754
  - GiNaC::Kronecker\_dtau\_kernel, 779
  - GiNaC::Kronecker\_dz\_kernel, 788
  - GiNaC::matrix, 807
  - GiNaC::modular\_form\_kernel, 836
  - GiNaC::multiple\_polylog\_kernel, 896
  - GiNaC::power, 988
  - GiNaC::pseries, 1036

- GiNaC::relational, [1072](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1121](#)
  - GiNaC::user\_defined\_kernel, [1225](#)
- normal
  - GiNaC, [120](#)
  - GiNaC::add, [320](#)
  - GiNaC::basic, [361](#)
  - GiNaC::ex, [567](#)
  - GiNaC::mul, [850](#)
  - GiNaC::numeric, [923](#)
  - GiNaC::power, [992](#)
  - GiNaC::pseries, [1039](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1126](#)
  - GiNaC::symbol, [1166](#)
- normal.cpp, [1323](#)
  - FAST\_COMPARE, [1325](#)
  - STATISTICS, [1325](#)
  - USE\_REMEMBER, [1325](#)
  - USE\_TRIAL\_DIVISION, [1325](#)
- normal.h, [1325](#)
- not\_equal
  - GiNaC::relational, [1071](#)
- not\_shareable
  - GiNaC::status\_flags, [1114](#)
- not\_symmetric
  - GiNaC, [265](#)
- nparams
  - GiNaC::function\_options, [687](#)
- nqexand\_transformer
  - GiNaC::Gt, [710](#)
- num
  - GiNaC::symminfo, [1185](#)
- num\_expressions
  - GiNaC::archive, [330](#)
- number
  - GiNaC::constant, [453](#)
- number\_of\_type
  - GiNaC, [136](#)
- numer
  - GiNaC, [119](#), [249](#)
  - GiNaC::ex, [569](#)
  - GiNaC::numeric, [937](#)
- numer\_denom
  - GiNaC, [119](#)
  - GiNaC::ex, [569](#)
- numeric
  - GiNaC::info\_flags, [746](#)
  - GiNaC::numeric, [919](#), [920](#)
- numeric.cpp, [1327](#)
- numeric.h, [1330](#)
- Nv
  - GiNaC::multi\_iterator\_counter\_indv< T >, [866](#)
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, [877](#)
- o
  - GiNaC::relational, [1077](#)
- obj
  - GiNaC::structure< T, ComparisonPolicy >, [1132](#)
- odd
  - GiNaC::info\_flags, [747](#)
- op
  - GiNaC, [123](#)
  - GiNaC::basic, [355](#)
  - GiNaC::clifford, [406](#)
  - GiNaC::container< C >, [462](#)
  - GiNaC::Ebar\_kernel, [511](#)
  - GiNaC::Eisenstein\_h\_kernel, [520](#)
  - GiNaC::Eisenstein\_kernel, [531](#)
  - GiNaC::ELi\_kernel, [541](#)
  - GiNaC::ex, [559](#)
  - GiNaC::expairseq, [597](#)
  - GiNaC::idx, [720](#)
  - GiNaC::integral, [754](#)
  - GiNaC::Kronecker\_dtau\_kernel, [779](#)
  - GiNaC::Kronecker\_dz\_kernel, [788](#)
  - GiNaC::matrix, [807](#)
  - GiNaC::modular\_form\_kernel, [836](#)
  - GiNaC::multiple\_polylog\_kernel, [896](#)
  - GiNaC::power, [988](#)
  - GiNaC::pseries, [1036](#)
  - GiNaC::relational, [1073](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1121](#)
  - GiNaC::user\_defined\_kernel, [1225](#)
- operator long
  - GiNaC::\_numeric\_digits, [308](#)
- operator safe\_bool
  - GiNaC::relational, [1076](#)
- operator!
  - GiNaC::relational, [1076](#)
- operator!=
  - GiNaC, [255](#)
  - GiNaC::const\_iterator, [435](#)
  - GiNaC::const\_postorder\_iterator, [439](#)
  - GiNaC::const\_preorder\_iterator, [442](#)
  - GiNaC::internal::\_iter\_rep, [306](#)
  - GiNaC::numeric, [934](#)
  - GiNaC::ptr< T >, [1051](#), [1052](#)
  - GiNaC::return\_type\_t, [1087](#)
- operator<
  - GiNaC, [255](#)
  - GiNaC::const\_iterator, [435](#)
  - GiNaC::Gt\_detail, [302](#)
  - GiNaC::Gt\_detail::pathintegral\_term, [949](#)
  - GiNaC::numeric, [935](#)
  - GiNaC::return\_type\_t, [1087](#)
  - GiNaC::spmapkey, [1112](#)
  - GiNaC::sym\_desc, [1158](#)
- operator<<
  - GiNaC, [93](#), [116](#), [117](#), [245](#), [257](#), [279–281](#)
  - GiNaC::archive, [332](#)
  - GiNaC::archive\_node, [342](#)
  - GiNaC::basic\_multi\_iterator< T >, [387](#)
  - GiNaC::Gt\_detail, [302](#)
  - GiNaC::Gt\_detail::pathintegral\_term, [948](#)
  - GiNaC::multi\_iterator\_counter< T >, [862](#)
  - GiNaC::multi\_iterator\_counter\_indv< T >, [866](#)
  - GiNaC::multi\_iterator\_ordered< T >, [870](#)

- GiNaC::multi\_iterator\_ordered\_eq< T >, 873
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 877
- GiNaC::multi\_iterator\_permutation< T >, 882
- GiNaC::multi\_iterator\_shuffle< T >, 885
- GiNaC::multi\_iterator\_shuffle\_prime< T >, 889
- GiNaC::ptr< T >, 1053
- operator<=
  - GiNaC, 256
  - GiNaC::const\_iterator, 435
  - GiNaC::numeric, 935
- operator>
  - GiNaC, 256
  - GiNaC::const\_iterator, 435
  - GiNaC::numeric, 935
- operator>>
  - GiNaC, 93, 94, 257
  - GiNaC::archive, 332
  - GiNaC::archive\_node, 342
- operator>=
  - GiNaC, 256
  - GiNaC::const\_iterator, 435
  - GiNaC::numeric, 935
- operator()
  - GiNaC::basic\_multi\_iterator< T >, 386
  - GiNaC::derivative\_map\_function, 476
  - GiNaC::error\_and\_integral\_is\_less, 545
  - GiNaC::eval\_integ\_map\_function, 547
  - GiNaC::evalf\_map\_function, 548
  - GiNaC::evalm\_map\_function, 550
  - GiNaC::ex\_base\_is\_less, 582
  - GiNaC::ex\_is\_equal, 583
  - GiNaC::ex\_is\_less, 583
  - GiNaC::ex\_swap, 584
  - GiNaC::expair\_is\_less, 589
  - GiNaC::expair\_rest\_is\_less, 589
  - GiNaC::expair\_swap, 590
  - GiNaC::expand\_map\_function, 610
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, 1214
  - GiNaC::idx\_is\_equal\_ignore\_dim, 727
  - GiNaC::is\_not\_a\_clifford, 768
  - GiNaC::is\_summation\_idx, 769
  - GiNaC::map\_function, 799
  - GiNaC::matrix, 812
  - GiNaC::normal\_map\_function, 912
  - GiNaC::op0\_is\_equal, 940
  - GiNaC::pointer\_to\_map\_function, 954
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 956
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 958
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 961
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, 964
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, 967
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 970
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 973
  - GiNaC::print\_functor, 1013
  - GiNaC::print\_functor\_impl, 1014
  - GiNaC::print\_memfun\_handler< T, C >, 1018
  - GiNaC::print\_ptrfun\_handler< T, C >, 1021
  - GiNaC::sy\_is\_less, 1155
  - GiNaC::sy\_swap, 1156
  - GiNaC::symminfo\_is\_less\_by\_orig, 1186
  - GiNaC::symminfo\_is\_less\_by\_symmterm, 1186
  - GiNaC::terminfo\_is\_less, 1213
  - std::equal\_to< GiNaC::ex >, 543
  - std::hash< GiNaC::ex >, 713
  - std::less< GiNaC::ptr< T > >, 792
- operator+
  - GiNaC, 250, 251, 253
  - GiNaC::const\_iterator, 434, 436
- operator++
  - GiNaC, 253–255
  - GiNaC::basic\_multi\_iterator< T >, 386
  - GiNaC::const\_iterator, 433
  - GiNaC::const\_postorder\_iterator, 438
  - GiNaC::const\_preorder\_iterator, 441
  - GiNaC::multi\_iterator\_counter< T >, 861
  - GiNaC::multi\_iterator\_counter\_indv< T >, 865
  - GiNaC::multi\_iterator\_ordered< T >, 869
  - GiNaC::multi\_iterator\_ordered\_eq< T >, 873
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 876
  - GiNaC::multi\_iterator\_permutation< T >, 881
  - GiNaC::multi\_iterator\_shuffle< T >, 885
- operator+=
  - GiNaC, 251, 252
  - GiNaC::const\_iterator, 434
- operator-
  - GiNaC, 250, 251, 253
  - GiNaC::const\_iterator, 434, 436
- operator->
  - GiNaC::const\_iterator, 433
  - GiNaC::const\_postorder\_iterator, 438
  - GiNaC::const\_preorder\_iterator, 441
  - GiNaC::ptr< T >, 1051
  - GiNaC::structure< T, ComparisonPolicy >, 1131
- operator--
  - GiNaC, 253–255
  - GiNaC::const\_iterator, 434
- operator-=
  - GiNaC, 251, 252
  - GiNaC::const\_iterator, 434
- operator/
  - GiNaC, 250, 251
- operator/=
  - GiNaC, 252
- operator=
  - GiNaC::numeric\_digits, 308
  - GiNaC::archive\_node, 337
  - GiNaC::basic, 352



- GiNaC::numeric, [929](#), [930](#)
  - GiNaC::print\_functor, [1013](#)
  - GiNaC::ptr< T >, [1050](#)
- operator==
  - GiNaC, [255](#)
  - GiNaC::const\_iterator, [434](#)
  - GiNaC::const\_postorder\_iterator, [439](#)
  - GiNaC::const\_preorder\_iterator, [442](#)
  - GiNaC::internal::\_iter\_rep, [306](#)
  - GiNaC::numeric, [934](#)
  - GiNaC::ptr< T >, [1051](#), [1052](#)
  - GiNaC::return\_type\_t, [1087](#)
  - GiNaC::spmapkey, [1112](#)
- operator[]
  - GiNaC::basic, [355](#), [356](#)
  - GiNaC::basic\_multi\_iterator< T >, [385](#), [386](#)
  - GiNaC::const\_iterator, [433](#)
  - GiNaC::ex, [560](#), [561](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1121](#), [1122](#)
- operator\*
  - GiNaC, [250](#), [251](#)
  - GiNaC::const\_iterator, [433](#)
  - GiNaC::const\_postorder\_iterator, [438](#)
  - GiNaC::const\_preorder\_iterator, [441](#)
  - GiNaC::ptr< T >, [1050](#)
- operator\*=
  - GiNaC, [252](#)
- operators
  - GiNaC::relational, [1071](#)
- operators.cpp, [1333](#)
- operators.h, [1335](#)
- options
  - GiNaC::class\_info< OPT >, [393](#)
  - GiNaC::expand\_map\_function, [610](#)
  - GiNaC::print\_context, [1000](#)
- Order\_conjugate
  - GiNaC, [158](#)
- Order\_eval
  - GiNaC, [157](#)
- Order\_expl\_derivative
  - GiNaC, [158](#)
- Order\_imag\_part
  - GiNaC, [158](#)
- Order\_power
  - GiNaC, [158](#)
- Order\_real\_part
  - GiNaC, [158](#)
- Order\_series
  - GiNaC, [158](#)
- orig
  - GiNaC::symminfo, [1185](#)
  - GiNaC::terminfo, [1212](#)
- overall\_coeff
  - GiNaC::expairseq, [607](#)
- overflow
  - GiNaC::basic\_multi\_iterator< T >, [385](#)
- overloaded
- GiNaC::function\_options, [685](#)
- P
- GiNaC::modular\_form\_kernel, [838](#)
- p
  - GiNaC::ptr< T >, [1053](#)
- parameter\_set
  - GiNaC::fderivative, [629](#)
- paramset
  - GiNaC, [70](#)
- parent
  - GiNaC::class\_info< OPT >, [393](#)
- parent\_name
  - GiNaC::print\_context\_options, [1002](#)
  - GiNaC::registered\_class\_options, [1065](#)
- parents\_identified
  - GiNaC::class\_info< OPT >, [393](#)
- partition
  - GiNaC::partition\_generator, [943](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [945](#)
- partition\_generator
  - GiNaC::partition\_generator, [942](#)
- partition\_with\_zero\_parts\_generator
  - GiNaC::partition\_with\_zero\_parts\_generator, [945](#)
- pathintegral\_term
  - GiNaC::Gt\_detail::pathintegral\_term, [947](#)
- pathintegral\_term\_list
  - GiNaC::Gt\_detail::pathintegral\_term, [948](#)
- pattern\_is\_not\_product
  - GiNaC::subs\_options, [1154](#)
- pattern\_is\_product
  - GiNaC::subs\_options, [1154](#)
- pderivative
  - GiNaC::function, [647](#)
- permutation\_sign
  - GiNaC, [277](#), [278](#)
- permute\_free\_index\_to\_front
  - GiNaC, [111](#)
- Pi
  - GiNaC, [283](#)
- PiEvalf
  - GiNaC, [244](#)
- pivot
  - GiNaC::matrix, [820](#)
- point
  - GiNaC::pseries, [1046](#)
- pointer
  - GiNaC::const\_iterator, [432](#)
  - GiNaC::const\_postorder\_iterator, [437](#)
  - GiNaC::const\_preorder\_iterator, [440](#)
- pointer\_to\_map\_function
  - GiNaC::pointer\_to\_map\_function, [954](#)
- pointer\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [956](#)
- pointer\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [958](#)
- pointer\_to\_map\_function\_3args



- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 961
- pointer\_to\_member\_to\_map\_function
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, 964
- pointer\_to\_member\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg<power\_funcp\_11 C, T1 >, 967
- pointer\_to\_member\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 970
- pointer\_to\_member\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args<power\_funcp\_14 C, T1, T2, T3 >, 973
- pole\_error
  - GiNaC::pole\_error, 975
- polylog
  - GiNaC::Gt\_detail::pathintegral\_term, 949
- polynomial
  - GiNaC::factor\_options, 611
  - GiNaC::info\_flags, 747
- pop
  - GiNaC::Gt\_detail::pathintegral\_term\_list, 951
- pos\_sig
  - GiNaC::minkmetric, 830
  - GiNaC::tensepsilon, 1199
- posint
  - GiNaC::info\_flags, 747
- positive
  - GiNaC::domain, 504
  - GiNaC::info\_flags, 747
- possymbol
  - GiNaC::possymbol, 981
- postorder\_begin
  - GiNaC::ex, 557
- postorder\_end
  - GiNaC::ex, 557
- pow
  - GiNaC, 245, 260
  - GiNaC::matrix, 812
- power
  - GiNaC::add, 326
  - GiNaC::function, 648
  - GiNaC::Gt\_detail::pathintegral\_term, 949
  - GiNaC::mul, 858
  - GiNaC::ncmul, 910
  - GiNaC::numeric, 927
  - GiNaC::power, 987
- power.cpp, 1336
- power.h, 1337
- power\_const
  - GiNaC::pseries, 1044
- power\_dyn
  - GiNaC::numeric, 929
- power\_f
  - GiNaC::function\_options, 688
- power\_func
  - GiNaC::function\_options, 674–676, 681
- power\_funcp
  - GiNaC, 71
- power\_funcp\_1
  - GiNaC, 72
- power\_funcp\_10
  - GiNaC, 83
- power\_funcp\_11
  - GiNaC, 85
- power\_funcp\_12
  - GiNaC, 86
- power\_funcp\_13
  - GiNaC, 88
- power\_funcp\_14
  - GiNaC, 89
- power\_funcp\_2
  - GiNaC, 73
- power\_funcp\_3
  - GiNaC, 74
- power\_funcp\_4
  - GiNaC, 75
- power\_funcp\_5
  - GiNaC, 77
- power\_funcp\_6
  - GiNaC, 78
- power\_funcp\_7
  - GiNaC, 79
- power\_funcp\_8
  - GiNaC, 81
- power\_funcp\_9
  - GiNaC, 82
- power\_funcp\_exvector
  - GiNaC, 90
- power\_use\_exvector\_args
  - GiNaC::function\_options, 691
- precedence
  - GiNaC::add, 317
  - GiNaC::basic, 354
  - GiNaC::clifford, 402
  - GiNaC::container< C >, 461
  - GiNaC::expairseq, 596
  - GiNaC::function, 642
  - GiNaC::Gt, 700
  - GiNaC::indexed, 739
  - GiNaC::integral, 753
  - GiNaC::mul, 847
  - GiNaC::ncmul, 905
  - GiNaC::numeric, 920
  - GiNaC::power, 987
  - GiNaC::pseries, 1036
  - GiNaC::relational, 1072
  - GiNaC::structure< T, ComparisonPolicy >, 1120
- prefactor
  - GiNaC::Gt\_detail::pathintegral\_term, 949
- prem
  - GiNaC, 217
- preorder\_begin
  - GiNaC::ex, 556
- preorder\_end

- GiNaC::ex, [556](#)
- prepend
  - GiNaC::container< C >, [466](#)
- prime
  - GiNaC::info\_flags, [747](#)
- primitive\_dirichlet\_character
  - GiNaC, [200](#)
- primpart
  - GiNaC::ex, [571](#), [572](#)
- print
  - GiNaC::\_numeric\_digits, [308](#)
  - GiNaC::basic, [353](#)
  - GiNaC::ex, [558](#)
  - GiNaC::expair, [587](#)
  - GiNaC::fderivative, [625](#)
  - GiNaC::function, [641](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1120](#)
- print.cpp, [1338](#)
- print.h, [1338](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT, [1340](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE, [1340](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON, [1340](#)
  - GINAC\_IMPLEMENT\_PRINT\_CONTEXT, [1340](#)
- print\_add
  - GiNaC::add, [325](#)
- print\_cache
  - GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, [1214](#)
- print\_context
  - GiNaC::print\_context, [999](#)
- print\_context\_class\_info
  - GiNaC, [92](#)
- print\_context\_options
  - GiNaC::print\_context\_options, [1001](#)
- print\_csrc
  - GiNaC::print\_csrc, [1004](#)
- print\_csrc\_cl\_N
  - GiNaC::print\_csrc\_cl\_N, [1006](#)
- print\_csrc\_double
  - GiNaC::print\_csrc\_double, [1008](#)
- print\_csrc\_float
  - GiNaC::print\_csrc\_float, [1010](#)
- print\_dflt
  - GiNaC::print\_dflt, [1011](#)
- print\_dispatch
  - GiNaC::basic, [368](#)
- print\_dispatch\_table
  - GiNaC::function\_options, [689](#)
  - GiNaC::registered\_class\_options, [1065](#)
- print\_elements
  - GiNaC::matrix, [821](#)
- print\_func
  - GiNaC::function\_options, [682–684](#)
  - GiNaC::registered\_class\_options, [1064](#), [1065](#)
- print\_func< print\_context >
  - GiNaC, [132](#)
- print\_func< print\_dflt >
  - GiNaC, [97](#), [110](#), [271](#)
- print\_funcp
  - GiNaC, [71](#)
- print\_funcp\_1
  - GiNaC, [72](#)
- print\_funcp\_10
  - GiNaC, [84](#)
- print\_funcp\_11
  - GiNaC, [85](#)
- print\_funcp\_12
  - GiNaC, [86](#)
- print\_funcp\_13
  - GiNaC, [88](#)
- print\_funcp\_14
  - GiNaC, [89](#)
- print\_funcp\_2
  - GiNaC, [73](#)
- print\_funcp\_3
  - GiNaC, [74](#)
- print\_funcp\_4
  - GiNaC, [76](#)
- print\_funcp\_5
  - GiNaC, [77](#)
- print\_funcp\_6
  - GiNaC, [78](#)
- print\_funcp\_7
  - GiNaC, [80](#)
- print\_funcp\_8
  - GiNaC, [81](#)
- print\_funcp\_9
  - GiNaC, [82](#)
- print\_funcp\_exvector
  - GiNaC, [91](#)
- print\_functor
  - GiNaC::print\_functor, [1012](#), [1013](#)
- print\_index
  - GiNaC::idx, [725](#)
- print\_index\_dimensions
  - GiNaC::print\_options, [1019](#)
- print\_indexed
  - GiNaC::indexed, [743](#)
- print\_integer\_csrc
  - GiNaC, [228](#)
- print\_latex
  - GiNaC::print\_latex, [1016](#)
- print\_memfun\_handler
  - GiNaC::print\_memfun\_handler< T, C >, [1018](#)
- print\_numeric
  - GiNaC::numeric, [938](#)
- print\_operator
  - GiNaC, [263](#)
- print\_overall\_coeff
  - GiNaC::mul, [856](#)
- print\_power
  - GiNaC::power, [995](#)
- print\_ptrfun\_handler
  - GiNaC::print\_ptrfun\_handler< T, C >, [1021](#)

- print\_python
  - GiNaC::print\_python, [1023](#)
- print\_python\_repr
  - GiNaC::print\_python\_repr, [1025](#)
- print\_real\_cl\_N
  - GiNaC, [230](#)
- print\_real\_csrc
  - GiNaC, [229](#)
- print\_real\_number
  - GiNaC, [228](#)
- print\_series
  - GiNaC::pseries, [1044](#)
- print\_sym\_pow
  - GiNaC, [259](#)
- print\_tree
  - GiNaC::print\_tree, [1026](#)
- print\_use\_exvector\_args
  - GiNaC::function\_options, [691](#)
- printindices
  - GiNaC::indexed, [743](#)
- printpair
  - GiNaC::expairseq, [601](#)
- printraw
  - GiNaC::archive, [332](#)
  - GiNaC::archive\_node, [342](#)
- printseq
  - GiNaC::container< C >, [465](#)
  - GiNaC::expairseq, [601](#)
- product\_to\_exvector
  - GiNaC, [138](#)
- property
  - GiNaC::archive\_node::property, [1028](#)
- property\_info
  - GiNaC::archive\_node::property\_info, [1029](#)
- property\_type
  - GiNaC::archive\_node, [336](#)
- propinfovector
  - GiNaC::archive\_node, [336](#)
- props
  - GiNaC::archive\_node, [343](#)
- pseries
  - GiNaC::pseries, [1035](#), [1036](#)
- pseries.cpp, [1341](#)
- pseries.h, [1342](#)
- psi
  - GiNaC, [161](#), [238](#)
- psi1\_deriv
  - GiNaC, [170](#)
- psi1\_eval
  - GiNaC, [170](#)
- psi1\_evalf
  - GiNaC, [170](#)
- psi1\_series
  - GiNaC, [170](#)
- psi2\_deriv
  - GiNaC, [171](#)
- psi2\_eval
  - GiNaC, [171](#)
- psi2\_evalf
  - GiNaC, [170](#)
- psi2\_series
  - GiNaC, [171](#)
- ptr
  - GiNaC::pointer\_to\_map\_function, [954](#)
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [956](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [959](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [961](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, [964](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [967](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [970](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [973](#)
  - GiNaC::ptr< T >, [1050](#)
- ptr.h, [1342](#)
- PTYPE\_BOOL
  - GiNaC::archive\_node, [336](#)
- PTYPE\_NODE
  - GiNaC::archive\_node, [337](#)
- PTYPE\_STRING
  - GiNaC::archive\_node, [337](#)
- PTYPE\_UNSIGNED
  - GiNaC::archive\_node, [337](#)
- purely\_indefinite
  - GiNaC::status\_flags, [1114](#)
- python
  - GiNaC, [258](#)
- python\_repr
  - GiNaC, [258](#)
- q\_expansion\_modular\_form
  - GiNaC::Eisenstein\_h\_kernel, [522](#)
  - GiNaC::Eisenstein\_kernel, [532](#)
  - GiNaC::modular\_form\_kernel, [838](#)
- qExpand
  - GiNaC::Gt, [704](#)
- qExpand\_minOrder
  - GiNaC::Gt, [709](#)
- qExpand\_stepSize
  - GiNaC::Gt, [709](#)
- quo
  - GiNaC, [215](#)
- r
  - GiNaC::Eisenstein\_h\_kernel, [523](#)
- rank
  - GiNaC, [210](#)
  - GiNaC::matrix, [817](#)
- rational
  - GiNaC::info\_flags, [747](#)
- rational\_function
  - GiNaC::info\_flags, [747](#)

- rational\_polynomial
  - GiNaC::info\_flags, 747
- rbegin
  - GiNaC::container< C >, 468
- read\_archive
  - GiNaC::basic, 369
  - GiNaC::clifford, 403
  - GiNaC::color, 422
  - GiNaC::constant, 450
  - GiNaC::container< C >, 463
  - GiNaC::expairseq, 599
  - GiNaC::fderivative, 626
  - GiNaC::function, 645
  - GiNaC::Gt, 702
  - GiNaC::idx, 721
  - GiNaC::indexed, 740
  - GiNaC::integral, 756
  - GiNaC::matrix, 809
  - GiNaC::minkmetric, 829
  - GiNaC::numeric, 925
  - GiNaC::power, 993
  - GiNaC::pseries, 1041
  - GiNaC::relational, 1073
  - GiNaC::spinidx, 1101
  - GiNaC::symbol, 1168
  - GiNaC::symmetry, 1179
  - GiNaC::tensepsilon, 1198
  - GiNaC::varidx, 1233
  - GiNaC::wildcard, 1241
- read\_real\_float
  - GiNaC, 227
- read\_unsigned
  - GiNaC, 93
- real
  - GiNaC, 249
  - GiNaC::domain, 504
  - GiNaC::info\_flags, 747
  - GiNaC::numeric, 937
- real\_part
  - GiNaC, 117
  - GiNaC::add, 321
  - GiNaC::basic, 366
  - GiNaC::constant, 449
  - GiNaC::container< C >, 464
  - GiNaC::ex, 561
  - GiNaC::function, 644
  - GiNaC::indexed, 740
  - GiNaC::matrix, 809
  - GiNaC::mul, 850
  - GiNaC::ncmul, 908
  - GiNaC::numeric, 925
  - GiNaC::power, 993
  - GiNaC::pseries, 1040
  - GiNaC::realsymbol, 1059
  - GiNaC::symbol, 1167
- real\_part\_conjugate
  - GiNaC, 143
- real\_part\_eval
  - GiNaC, 143
- real\_part\_evalf
  - GiNaC, 143
- real\_part\_expl\_derivative
  - GiNaC, 144
- real\_part\_f
  - GiNaC::function\_options, 688
- real\_part\_func
  - GiNaC::function\_options, 664–666, 680
- real\_part\_funcp
  - GiNaC, 70
- real\_part\_funcp\_1
  - GiNaC, 71
- real\_part\_funcp\_10
  - GiNaC, 83
- real\_part\_funcp\_11
  - GiNaC, 84
- real\_part\_funcp\_12
  - GiNaC, 86
- real\_part\_funcp\_13
  - GiNaC, 87
- real\_part\_funcp\_14
  - GiNaC, 89
- real\_part\_funcp\_2
  - GiNaC, 72
- real\_part\_funcp\_3
  - GiNaC, 74
- real\_part\_funcp\_4
  - GiNaC, 75
- real\_part\_funcp\_5
  - GiNaC, 76
- real\_part\_funcp\_6
  - GiNaC, 77
- real\_part\_funcp\_7
  - GiNaC, 79
- real\_part\_funcp\_8
  - GiNaC, 80
- real\_part\_funcp\_9
  - GiNaC, 81
- real\_part\_funcp\_exvector
  - GiNaC, 90
- real\_part\_imag\_part
  - GiNaC, 144
- real\_part\_print\_latex
  - GiNaC, 143
- real\_part\_real\_part
  - GiNaC, 143
- real\_part\_use\_exvector\_args
  - GiNaC::function\_options, 690
- really\_subs\_idx
  - GiNaC::subs\_options, 1154
- realsymbol
  - GiNaC::realsymbol, 1058, 1059
- recombine\_pair\_to\_ex
  - GiNaC::add, 324
  - GiNaC::expairseq, 602
  - GiNaC::mul, 854
- reduced\_matrix

- GiNaC, 207
- reeval\_ncmul
  - GiNaC, 212
  - GiNaC::ncmul, 910
- refcount
  - GiNaC::refcounted, 1063
- refcounted
  - GiNaC::refcounted, 1062
- reference
  - GiNaC::const\_iterator, 432
  - GiNaC::const\_postorder\_iterator, 438
  - GiNaC::const\_preorder\_iterator, 441
- REGISTER\_FUNCTION
  - function.h, 1289
  - GiNaC, 142, 144, 145, 148, 149, 151–154, 156, 157, 159, 163, 165, 167–169, 173, 174, 176, 179, 181, 183, 184, 186–190, 192, 193, 195–198
- register\_new
  - GiNaC::function, 648
- registered\_class\_info
  - GiNaC, 92
- registered\_class\_options
  - GiNaC::registered\_class\_options, 1064
- registered\_functions
  - GiNaC::function, 647
- registrar.cpp, 1343
- registrar.h, 1343
  - GINAC\_DECLARE\_REGISTERED\_CLASS, 1345
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON, 1344
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS, 1344
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS, 1345
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, 1345
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T, 1346
- regularise
  - GiNaC::Gt, 703
- relation
  - GiNaC::info\_flags, 747
- relation\_equal
  - GiNaC::info\_flags, 747
- relation\_greater
  - GiNaC::info\_flags, 747
- relation\_greater\_or\_equal
  - GiNaC::info\_flags, 747
- relation\_less
  - GiNaC::info\_flags, 747
- relation\_less\_or\_equal
  - GiNaC::info\_flags, 747
- relation\_not\_equal
  - GiNaC::info\_flags, 747
- relational
  - GiNaC::relational, 1072
- relational.cpp, 1346
- relational.h, 1347
- relative\_integration\_error
  - GiNaC::integral, 758
- rem
  - GiNaC, 216
- remember
  - GiNaC::function\_options, 684
- remember.cpp, 1347
- remember.h, 1348
- remember\_assoc\_size
  - GiNaC::function\_options, 690
- remember\_size
  - GiNaC::function\_options, 690
- remember\_strategy
  - GiNaC::function\_options, 690
- GiNaC::remember\_table, 1081
- GiNaC::remember\_table\_list, 1086
- remember\_table
  - GiNaC::remember\_table, 1079
- remember\_table\_entry
  - GiNaC::function, 649
  - GiNaC::remember\_table\_entry, 1083
- remember\_table\_list
  - GiNaC::remember\_table\_list, 1086
- remember\_tables
  - GiNaC::remember\_table, 1080
- remove\_all
  - GiNaC::container< C >, 467
- remove\_dirac\_ONE
  - GiNaC, 104
- remove\_first
  - GiNaC::container< C >, 466
- remove\_last
  - GiNaC::container< C >, 466
- remove\_reference
  - GiNaC::refcounted, 1062
- rename\_dummy\_indices
  - GiNaC, 137
- rename\_dummy\_indices\_uniquely
  - GiNaC, 139, 140
- rend
  - GiNaC::container< C >, 468
- replace\_contr\_index
  - GiNaC::tensor, 1210
- replace\_dim
  - GiNaC::idx, 724
- replace\_with\_symbol
  - GiNaC, 225
- reposition\_dummy\_indices
  - GiNaC, 137
  - GiNaC::indexed, 745
- representation\_label
  - GiNaC::clifford, 407
  - GiNaC::color, 424
- reserve
  - GiNaC::container\_storage< C >, 471
- rest
  - GiNaC::expair, 588

- result
  - GiNaC::remember\_table\_entry, 1084
- result\_type
  - GiNaC::map\_function, 799
- resultant
  - GiNaC, 227
- return\_type
  - GiNaC::add, 322
  - GiNaC::basic, 365
  - GiNaC::clifford, 404
  - GiNaC::color, 424
  - GiNaC::ex, 577
  - GiNaC::expairseq, 600
  - GiNaC::fail, 616
  - GiNaC::function, 646
  - GiNaC::function\_options, 689
  - GiNaC::indexed, 741
  - GiNaC::integral, 755
  - GiNaC::matrix, 810
  - GiNaC::minkmetric, 829
  - GiNaC::mul, 853
  - GiNaC::ncmul, 908
  - GiNaC::power, 994
  - GiNaC::relational, 1074
  - GiNaC::structure< T, ComparisonPolicy >, 1130
  - GiNaC::su3d, 1137
  - GiNaC::su3f, 1143
  - GiNaC::tensdelta, 1192
  - GiNaC::tensepsilon, 1199
  - GiNaC::tensmetric, 1205
  - GiNaC::tensor, 1210
- return\_type\_tinfo
  - GiNaC::add, 323
  - GiNaC::basic, 366
  - GiNaC::clifford, 404
  - GiNaC::color, 424
  - GiNaC::ex, 577
  - GiNaC::function, 647
  - GiNaC::function\_options, 689
  - GiNaC::indexed, 742
  - GiNaC::integral, 756
  - GiNaC::mul, 853
  - GiNaC::ncmul, 908
  - GiNaC::power, 994
  - GiNaC::relational, 1075
  - GiNaC::structure< T, ComparisonPolicy >, 1131
- rh
  - GiNaC::relational, 1077
- rhs
  - GiNaC, 123
  - GiNaC::ex, 561
  - GiNaC::relational, 1076
- rl
  - GiNaC::return\_type\_t, 1088
- root
  - GiNaC::archive::archived\_ex, 346
- rotate\_left
  - GiNaC, 277
- row
  - GiNaC::matrix, 822
- rows
  - GiNaC, 209
  - GiNaC::matrix, 810
- s
  - GiNaC::const\_postorder\_iterator, 439
  - GiNaC::const\_preorder\_iterator, 442
  - GiNaC::derivative\_map\_function, 476
  - GiNaC::Eisenstein\_h\_kernel, 523
  - GiNaC::print\_context, 1000
  - GiNaC::symbolset, 1173
- S\_deriv
  - GiNaC, 174
- S\_eval
  - GiNaC, 173
- S\_evalf
  - GiNaC, 173
- S\_print\_latex
  - GiNaC, 174
- S\_series
  - GiNaC, 174
- safe\_bool
  - GiNaC::relational, 1071
- same\_metric
  - GiNaC::clifford, 405
- scalar\_mul\_indexed
  - GiNaC::basic, 364
  - GiNaC::matrix, 808
  - GiNaC::structure< T, ComparisonPolicy >, 1128
- seq
  - GiNaC::container\_storage< C >, 472
  - GiNaC::expairseq, 607
  - GiNaC::pseries, 1046
  - GiNaC::remember\_table\_entry, 1084
- serial
  - GiNaC::constant, 453
  - GiNaC::function, 649
  - GiNaC::G2\_SERIAL, 692
  - GiNaC::G3\_SERIAL, 693
  - GiNaC::iterated\_integral2\_SERIAL, 769
  - GiNaC::iterated\_integral3\_SERIAL, 770
  - GiNaC::psi1\_SERIAL, 1047
  - GiNaC::psi2\_SERIAL, 1048
  - GiNaC::symbol, 1171
  - GiNaC::zeta1\_SERIAL, 1243
  - GiNaC::zeta2\_SERIAL, 1244
- series
  - GiNaC, 121
  - GiNaC::add, 319
  - GiNaC::basic, 361
  - GiNaC::Eisenstein\_h\_kernel, 520
  - GiNaC::Eisenstein\_kernel, 530
  - GiNaC::ex, 567
  - GiNaC::fderivative, 625
  - GiNaC::function, 643
  - GiNaC::integral, 757
  - GiNaC::integration\_kernel, 764

- GiNaC::modular\_form\_kernel, 836
  - GiNaC::mul, 850
  - GiNaC::power, 991
  - GiNaC::pseries, 1038
  - GiNaC::structure< T, ComparisonPolicy >, 1125
  - GiNaC::symbol, 1166
- series\_coeff
  - GiNaC::integration\_kernel, 767
- series\_coeff\_impl
  - GiNaC::basic\_log\_kernel, 381
  - GiNaC::Ebar\_kernel, 512
  - GiNaC::ELi\_kernel, 542
  - GiNaC::integration\_kernel, 766
  - GiNaC::Kronecker\_dtau\_kernel, 780
  - GiNaC::Kronecker\_dz\_kernel, 789
  - GiNaC::multiple\_polylog\_kernel, 896
- series\_f
  - GiNaC::function\_options, 689
- series\_func
  - GiNaC::function\_options, 676–678, 681
- series\_funcp
  - GiNaC, 71
- series\_funcp\_1
  - GiNaC, 72
- series\_funcp\_10
  - GiNaC, 83
- series\_funcp\_11
  - GiNaC, 85
- series\_funcp\_12
  - GiNaC, 86
- series\_funcp\_13
  - GiNaC, 88
- series\_funcp\_14
  - GiNaC, 89
- series\_funcp\_2
  - GiNaC, 73
- series\_funcp\_3
  - GiNaC, 74
- series\_funcp\_4
  - GiNaC, 75
- series\_funcp\_5
  - GiNaC, 77
- series\_funcp\_6
  - GiNaC, 78
- series\_funcp\_7
  - GiNaC, 79
- series\_funcp\_8
  - GiNaC, 81
- series\_funcp\_9
  - GiNaC, 82
- series\_funcp\_exvector
  - GiNaC, 91
- series\_to\_poly
  - GiNaC, 261
- series\_use\_exvector\_args
  - GiNaC::function\_options, 691
- series\_vec
  - GiNaC::integration\_kernel, 768
- set
  - GiNaC::matrix, 814
- set\_cache\_step
  - GiNaC::integration\_kernel, 766
- set\_name
  - GiNaC::function\_options, 658
  - GiNaC::symbol, 1169
- set\_print\_context
  - GiNaC, 257
- set\_print\_func
  - GiNaC, 262
  - GiNaC::function\_options, 686
  - GiNaC::registered\_class\_options, 1065
- set\_print\_options
  - GiNaC, 257
- set\_refcount
  - GiNaC::refcounted, 1062
- set\_return\_type
  - GiNaC::function\_options, 684
- set\_symmetry
  - GiNaC::function\_options, 685
- set\_TeX\_name
  - GiNaC::symbol, 1169
- set\_type
  - GiNaC::symmetry, 1180
- setArgs
  - GiNaC::Gt, 700
- setflag
  - GiNaC::basic, 372
- shaker\_sort
  - GiNaC, 278
- share
  - GiNaC::ex, 579
- shift\_exponents
  - GiNaC::pseries, 1044
- show\_statistics
  - GiNaC::remember\_table, 1080
- simplify\_indexed
  - GiNaC, 122, 138
  - GiNaC::ex, 573, 574
  - GiNaC::indexed, 745
- simplify\_indexed\_product
  - GiNaC, 138
  - GiNaC::indexed, 745
- sin
  - GiNaC, 231
- sin\_conjugate
  - GiNaC, 182
- sin\_deriv
  - GiNaC, 182
- sin\_eval
  - GiNaC, 181
- sin\_evalf
  - GiNaC, 181
- sin\_imag\_part
  - GiNaC, 182
- sin\_real\_part
  - GiNaC, 182

- sinh
  - GiNaC, 233
- sinh\_conjugate
  - GiNaC, 191
- sinh\_deriv
  - GiNaC, 191
- sinh\_eval
  - GiNaC, 191
- sinh\_evalf
  - GiNaC, 190
- sinh\_imag\_part
  - GiNaC, 191
- sinh\_real\_part
  - GiNaC, 191
- size
  - GiNaC::basic\_multi\_iterator< T >, 385
  - GiNaC::Gt\_detail::pathintegral\_term\_list, 951
- smod
  - GiNaC, 241
  - GiNaC::add, 320
  - GiNaC::basic, 363
  - GiNaC::ex, 573
  - GiNaC::mul, 851
  - GiNaC::numeric, 924
  - GiNaC::structure< T, ComparisonPolicy >, 1127
- solve
  - GiNaC::matrix, 817
- sort
  - GiNaC::container< C >, 467
- sort\_
  - GiNaC::container< C >, 465
- spinidx
  - GiNaC::spinidx, 1100
- spinor\_metric
  - GiNaC, 274
- split\_ex\_to\_pair
  - GiNaC::add, 323
  - GiNaC::expairseq, 601
  - GiNaC::mul, 853
- spm
  - GiNaC::scalar\_products, 1091
- spmmap
  - GiNaC, 91
- spmmapkey
  - GiNaC::spmmapkey, 1112
- sprem
  - GiNaC, 217
- sqrfree
  - GiNaC, 223
- sqrfree\_parfrac
  - GiNaC, 224
- sqrfree\_yun
  - GiNaC, 223
- sqrt
  - GiNaC, 244, 260
- sr\_gcd
  - GiNaC, 219
- STATISTICS
  - normal.cpp, 1325
- std, 302
  - swap, 303
- std::equal\_to< GiNaC::ex >, 543
  - operator(), 543
- std::hash< GiNaC::ex >, 713
  - operator(), 713
- std::less< GiNaC::ptr< T > >, 792
  - operator(), 792
- std::less< ptr< T > >
  - GiNaC::ptr< T >, 1052
- step
  - GiNaC, 246
  - GiNaC::numeric, 930
- step\_conjugate
  - GiNaC, 149
- step\_eval
  - GiNaC, 148
- step\_evalf
  - GiNaC, 148
- step\_imag\_part
  - GiNaC, 149
- step\_real\_part
  - GiNaC, 149
- step\_series
  - GiNaC, 148
- STLT
  - GiNaC::container< C >, 460
  - GiNaC::container\_storage< C >, 470
- store\_remember\_table
  - GiNaC::function, 648
- struct\_compare
  - GiNaC::compare\_all\_equal< T >, 425
  - GiNaC::compare\_bitwise< T >, 426
  - GiNaC::compare\_std\_less< T >, 427
- struct\_is\_equal
  - GiNaC::compare\_all\_equal< T >, 425
  - GiNaC::compare\_bitwise< T >, 426
  - GiNaC::compare\_std\_less< T >, 427
- structure
  - GiNaC::structure< T, ComparisonPolicy >, 1119
- structure< T, CP >::reg\_info
  - GiNaC, 285
- structure.h, 1348
- sub
  - GiNaC::matrix, 811
  - GiNaC::numeric, 927
- sub\_dyn
  - GiNaC::numeric, 928
- sub\_matrix
  - GiNaC, 208
- subs
  - GiNaC, 124
  - GiNaC::basic, 357
  - GiNaC::clifford, 406
  - GiNaC::container< C >, 462
  - GiNaC::ex, 563
  - GiNaC::expairseq, 598



- GiNaC::Gt, 701
- GiNaC::idx, 721
- GiNaC::matrix, 807
- GiNaC::numeric, 922
- GiNaC::power, 991
- GiNaC::pseries, 1038
- GiNaC::relational, 1073
- GiNaC::structure< T, ComparisonPolicy >, 1123
- GiNaC::symbol, 1166
- subs\_algebraic
  - GiNaC::subs\_options, 1154
- subs\_no\_pattern
  - GiNaC::subs\_options, 1154
- subs\_one\_level
  - GiNaC::basic, 369
- subschildren
  - GiNaC::container< C >, 469
  - GiNaC::expairseq, 607
- subsvvalue
  - GiNaC, 198
- successful\_hits
  - GiNaC::remember\_table\_entry, 1084
- sufficiently\_accurate
  - GiNaC::lanczos\_coeffs, 791
- suppress\_branchcut
  - GiNaC::series\_options, 1092
- swap
  - GiNaC, 124, 129
  - GiNaC::ex, 556
  - GiNaC::expair, 587
  - GiNaC::ptr< T >, 1051
  - std, 303
- swapped
  - GiNaC::sy\_swap, 1156
- sy\_anti
  - GiNaC, 269, 270
- sy\_cycl
  - GiNaC, 270
- sy\_is\_less
  - GiNaC::sy\_is\_less, 1155
  - GiNaC::symmetry, 1182
- sy\_none
  - GiNaC, 268
- sy\_swap
  - GiNaC::sy\_swap, 1156
  - GiNaC::symmetry, 1182
- sy\_symm
  - GiNaC, 269
- sym
  - GiNaC::sym\_desc, 1159
- sym\_desc
  - GiNaC::sym\_desc, 1158
- sym\_desc\_vec
  - GiNaC, 92
- symbol
  - GiNaC::info\_flags, 747
  - GiNaC::symbol, 1165
- symbol.cpp, 1349
- symbol.h, 1350
- symbolic\_matrix
  - GiNaC, 207, 210
- symbolset
  - GiNaC::symbolset, 1172
- symm
  - GiNaC, 267
  - GiNaC::terminfo, 1212
- symmetric
  - GiNaC::symmetry, 1178
- symmetric2
  - GiNaC, 266
- symmetric3
  - GiNaC, 266
- symmetric4
  - GiNaC, 266
- symmetrize
  - GiNaC, 122, 267, 271
  - GiNaC::ex, 575, 576
- symmetrize\_cyclic
  - GiNaC, 123, 268, 271
  - GiNaC::ex, 576
- symmetry
  - GiNaC::symmetry, 1178
- symmetry.cpp, 1351
- symmetry.h, 1352
- symmetry\_type
  - GiNaC::symmetry, 1178
- symminfo
  - GiNaC::symminfo, 1185
- symmterm
  - GiNaC::symminfo, 1185
- symtree
  - GiNaC::function\_options, 692
  - GiNaC::indexed, 746
- synthesize\_func
  - GiNaC, 68
- table\_size
  - GiNaC::remember\_table, 1081
- tan
  - GiNaC, 231
- tan\_conjugate
  - GiNaC, 185
- tan\_deriv
  - GiNaC, 185
- tan\_eval
  - GiNaC, 184
- tan\_evalf
  - GiNaC, 184
- tan\_imag\_part
  - GiNaC, 185
- tan\_real\_part
  - GiNaC, 185
- tan\_series
  - GiNaC, 185
- tanh
  - GiNaC, 234
- tanh\_conjugate

- GiNaC, [194](#)
- tanh\_deriv
  - GiNaC, [194](#)
- tanh\_eval
  - GiNaC, [193](#)
- tanh\_evalf
  - GiNaC, [193](#)
- tanh\_imag\_part
  - GiNaC, [194](#)
- tanh\_real\_part
  - GiNaC, [194](#)
- tanh\_series
  - GiNaC, [194](#)
- tau
  - GiNaC::Gt, [710](#)
  - GiNaC::Kronecker\_dz\_kernel, [790](#)
- tauToFundamental
  - GiNaC::Gt, [703](#)
- tcoeff
  - GiNaC::ex, [566](#)
- tensepsilon
  - GiNaC::tensepsilon, [1197](#)
- tensor
  - GiNaC, [283](#)
- tensor.cpp, [1353](#)
- tensor.h, [1354](#)
- terminfo
  - GiNaC::terminfo, [1212](#)
- terms
  - GiNaC::Gt\_detail::pathintegral\_term\_list, [952](#)
- test
  - GiNaC::has\_distance< T >, [712](#)
- test\_and\_set\_nparams
  - GiNaC::function\_options, [685](#)
- TEST\_PERMUTATION
  - color.cpp, [1259](#)
- TeX\_name
  - GiNaC::constant, [453](#)
  - GiNaC::function\_options, [687](#)
  - GiNaC::symbol, [1171](#)
- tgamma
  - GiNaC, [237](#)
- tgamma\_conjugate
  - GiNaC, [168](#)
- tgamma\_deriv
  - GiNaC, [168](#)
- tgamma\_eval
  - GiNaC, [167](#)
- tgamma\_evalf
  - GiNaC, [167](#)
- tgamma\_series
  - GiNaC, [168](#)
- thiscontainer
  - GiNaC::clifford, [404](#)
  - GiNaC::color, [423](#)
  - GiNaC::container< C >, [464](#), [465](#)
  - GiNaC::fderivative, [626](#)
  - GiNaC::function, [643](#), [644](#)
  - GiNaC::indexed, [741](#)
  - GiNaC::ncmul, [907](#)
- thisexpairseq
  - GiNaC::add, [323](#)
  - GiNaC::expairseq, [600](#), [601](#)
  - GiNaC::mul, [853](#)
- tinfo
  - GiNaC::return\_type\_t, [1088](#)
- tinfo\_key
  - GiNaC::registered\_class\_options, [1065](#)
- to\_cl\_N
  - GiNaC::numeric, [936](#)
- to\_double
  - GiNaC, [248](#)
  - GiNaC::numeric, [936](#)
- to\_int
  - GiNaC, [248](#)
  - GiNaC::numeric, [936](#)
- to\_integer
  - GiNaC::Gt, [708](#)
- to\_long
  - GiNaC, [248](#)
  - GiNaC::numeric, [936](#)
- to\_numeric
  - GiNaC::Gt, [708](#)
- to\_polynomial
  - GiNaC, [120](#)
  - GiNaC::basic, [362](#)
  - GiNaC::ex, [568](#)
  - GiNaC::expairseq, [598](#)
  - GiNaC::numeric, [923](#)
  - GiNaC::power, [992](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1126](#)
  - GiNaC::symbol, [1167](#)
- to\_rational
  - GiNaC, [120](#)
  - GiNaC::basic, [362](#)
  - GiNaC::ex, [568](#)
  - GiNaC::expairseq, [597](#)
  - GiNaC::numeric, [923](#)
  - GiNaC::power, [992](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1126](#)
  - GiNaC::symbol, [1166](#)
- toggle\_dot
  - GiNaC::spinidx, [1103](#)
- toggle\_variance
  - GiNaC::varidx, [1235](#)
- toggle\_variance\_dot
  - GiNaC::spinidx, [1103](#)
- too\_late
  - GiNaC::\_numeric\_digits, [309](#)
- trace
  - GiNaC, [209](#)
  - GiNaC::matrix, [815](#)
- trace\_string
  - GiNaC, [102](#)
- TransformExpressionWithCache

- GiNaC::Gt\_detail::TransformExpressionWithCache< Type >, 1214
- transpose
  - GiNaC, 209
  - GiNaC::matrix, 814
- traverse
  - GiNaC::ex, 564
- traverse\_postorder
  - GiNaC::ex, 564
- traverse\_preorder
  - GiNaC::ex, 564
- tree
  - GiNaC, 258
- tree\_node
  - GiNaC::class\_info< OPT >::tree\_node, 1217
- trig\_info
  - GiNaC, 182
- trivial
  - GiNaC::composition\_generator, 429
- tryfactsubs
  - GiNaC, 211
- type
  - GiNaC::archive\_node::property, 1028
  - GiNaC::archive\_node::property\_info, 1030
  - GiNaC::symmetry, 1183
- uintvector
  - GiNaC, 91
- unarch\_map
  - GiNaC::unarchive\_table\_t, 1219
- unarch\_table\_instance
  - GiNaC, 282
- unarchive
  - GiNaC::archive\_node, 341
- unarchive\_ex
  - GiNaC::archive, 329, 330
- unarchive\_map\_t
  - GiNaC, 68
- unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 1218
- unatomize
  - GiNaC::archive, 332
- unique
  - GiNaC::container< C >, 467
- unique\_
  - GiNaC::container< C >, 466, 469
- unit
  - GiNaC::ex, 570
- unit\_matrix
  - GiNaC, 207, 210
- unitcontprim
  - GiNaC::ex, 572
- unlikely
  - compiler.h, 1262
- unlink\_ex
  - GiNaC, 129
- unsignedvector
  - GiNaC, 91
- USE\_REMEMBER
  - normal.cpp, 1325
- use\_remember
  - GiNaC::function\_options, 690
- use\_return\_type
  - GiNaC::function\_options, 689
- USE\_SAME\_DEGREE\_FACTOR
  - factor.cpp, 1274
- use\_sr\_gcd
  - GiNaC::gcd\_options, 694
- USE\_TRIAL\_DIVISION
  - normal.cpp, 1325
- usecount
  - GiNaC::unarchive\_table\_t, 1219
- used\_indices
  - GiNaC::make\_flat\_inserter, 796
- user\_defined\_kernel
  - GiNaC::user\_defined\_kernel, 1225
- uses\_Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, 521
  - GiNaC::Eisenstein\_kernel, 532
  - GiNaC::integration\_kernel, 765
  - GiNaC::modular\_form\_kernel, 837
  - GiNaC::user\_defined\_kernel, 1226
- utils.cpp, 1355
- utils.h, 1358
  - DEFAULT\_COMPARE, 1359
  - DEFAULT\_CTOR, 1359
  - DEFAULT\_PRINT, 1359
  - DEFAULT\_PRINT\_LATEX, 1360
- utils\_multi\_iterator.h, 1360
- v
  - GiNaC::basic\_multi\_iterator< T >, 387
  - GiNaC::sy\_is\_less, 1155
  - GiNaC::sy\_swap, 1156
- v1
  - GiNaC::spmapkey, 1112
- v2
  - GiNaC::spmapkey, 1112
- v\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, 885
- v\_orig
  - GiNaC::multi\_iterator\_shuffle< T >, 886
- validate
  - GiNaC::indexed, 744
  - GiNaC::symmetry, 1180
- value
  - GiNaC::archive\_node::property, 1028
  - GiNaC::composition\_generator::coolmulti::element, 534
  - GiNaC::has\_distance< T >, 711
  - GiNaC::idx, 726
  - GiNaC::numeric, 940
- value\_type
  - GiNaC::const\_iterator, 432
  - GiNaC::const\_postorder\_iterator, 437
  - GiNaC::const\_preorder\_iterator, 440
- var
  - GiNaC::pseries, 1046

- varidx
  - GiNaC::varidx, [1233](#)
- version.h, [1361](#)
  - GINAC\_LT\_AGE, [1362](#)
  - GINAC\_LT\_CURRENT, [1362](#)
  - GINAC\_LT\_REVISION, [1362](#)
  - GINACLIB\_ARCHIVE\_AGE, [1363](#)
  - GINACLIB\_ARCHIVE\_VERSION, [1363](#)
  - GINACLIB\_MAJOR\_VERSION, [1362](#)
  - GINACLIB\_MICRO\_VERSION, [1362](#)
  - GINACLIB\_MINOR\_VERSION, [1362](#)
  - GINACLIB\_STR, [1363](#)
  - GINACLIB\_STR\_HELPER, [1363](#)
  - GINACLIB\_VERSION, [1363](#)
- version\_major
  - GiNaC, [286](#)
- version\_micro
  - GiNaC, [286](#)
- version\_minor
  - GiNaC, [286](#)
- wild
  - GiNaC, [282](#)
- wildcard
  - GiNaC::wildcard, [1240](#)
- wildcard.cpp, [1363](#)
- wildcard.h, [1364](#)
- write\_real\_float
  - GiNaC, [228](#)
- write\_unsigned
  - GiNaC, [93](#)
- x
  - GiNaC::basic\_partition\_generator::mpartition2, [840](#)
  - GiNaC::Ebar\_kernel, [513](#)
  - GiNaC::ELi\_kernel, [543](#)
  - GiNaC::integral, [758](#)
  - GiNaC::user\_defined\_kernel, [1226](#)
- y
  - GiNaC::Ebar\_kernel, [513](#)
  - GiNaC::ELi\_kernel, [543](#)
- yes\_type
  - GiNaC::has\_distance< T >, [711](#)
- z
  - GiNaC::Gt, [710](#)
  - GiNaC::Kronecker\_dtau\_kernel, [781](#)
  - GiNaC::multiple\_polylog\_kernel, [897](#)
- z\_j
  - GiNaC::Kronecker\_dz\_kernel, [790](#)
- zeta
  - GiNaC, [160](#), [236](#)
- zeta1\_deriv
  - GiNaC, [176](#)
- zeta1\_eval
  - GiNaC, [176](#)
- zeta1\_evalf
  - GiNaC, [176](#)
- zeta1\_print\_latex
  - GiNaC, [176](#)
- zeta2\_deriv
  - GiNaC, [177](#)
- zeta2\_eval
  - GiNaC, [177](#)
- zeta2\_evalf
  - GiNaC, [177](#)
- zeta2\_print\_latex
  - GiNaC, [177](#)
- zetaderiv\_deriv
  - GiNaC, [154](#)
- zetaderiv\_eval
  - GiNaC, [154](#)
- zi
  - GiNaC::Gt::kernel, [773](#)
- zisToFundamental
  - GiNaC::Gt, [703](#)