INTERNATIONAL STANDARD

ISO/IEC 14882

Third edition
2011-09-01

Information technology — Programming languages — C++

Technologies de l'information — Langages de programmation — C++

Reference number
ISO/IEC 14882:2011(E)

© ISO/IEC 2011

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2011

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Contents iii

List of Tables xi

List of Figures xv

1 General 1
- 1.1 Scope . . . 1
- 1.2 Normative references . . . 1
- 1.3 Terms and definitions . . . 2
- 1.4 Implementation compliance . . . 5
- 1.5 Structure of this International Standard . . . 6
- 1.6 Syntax notation . . . 6
- 1.7 The C++ memory model . . . 7
- 1.8 The C++ object model . . . 7
- 1.9 Program execution . . . 8
- 1.10 Multi-threaded executions and data races . . . 12
- 1.11 Acknowledgments . . . 16

2 Lexical conventions 17
- 2.1 Separate translation . . . 17
- 2.2 Phases of translation . . . 17
- 2.3 Character sets . . . 18
- 2.4 Trigraph sequences . . . 19
- 2.5 Preprocessing tokens . . . 20
- 2.6 Alternative tokens . . . 21
- 2.7 Tokens . . . 21
- 2.8 Comments . . . 21
- 2.9 Header names . . . 22
- 2.10 Preprocessing numbers . . . 22
- 2.11 Identifiers . . . 22
- 2.12 Keywords . . . 23
- 2.13 Operators and punctuators . . . 24
- 2.14 Literals . . . 24

3 Basic concepts 34
- 3.1 Declarations and definitions . . . 34
- 3.2 One definition rule . . . 36
- 3.3 Scope . . . 38
- 3.4 Name lookup . . . 45
- 3.5 Program and linkage . . . 59
- 3.6 Start and termination . . . 62
- 3.7 Storage duration . . . 65
- 3.8 Object lifetime . . . 69
- 3.9 Types . . . 72
- 3.10 Lvalues and rvalues . . . 78

© ISO/IEC 2011 – All rights reserved

3.11 Alignment . . . 80

4 Standard conversions **81**
4.1 Lvalue-to-rvalue conversion . . . 82
4.2 Array-to-pointer conversion . . . 82
4.3 Function-to-pointer conversion . . . 82
4.4 Qualification conversions . . . 82
4.5 Integral promotions . . . 83
4.6 Floating point promotion . . . 84
4.7 Integral conversions . . . 84
4.8 Floating point conversions . . . 84
4.9 Floating-integral conversions . . . 85
4.10 Pointer conversions . . . 85
4.11 Pointer to member conversions . . . 85
4.12 Boolean conversions . . . 86
4.13 Integer conversion rank . . . 86

5 Expressions **87**
5.1 Primary expressions . . . 89
5.2 Postfix expressions . . . 97
5.3 Unary expressions . . . 109
5.4 Explicit type conversion (cast notation) . . . 117
5.5 Pointer-to-member operators . . . 118
5.6 Multiplicative operators . . . 119
5.7 Additive operators . . . 119
5.8 Shift operators . . . 121
5.9 Relational operators . . . 121
5.10 Equality operators . . . 122
5.11 Bitwise AND operator . . . 123
5.12 Bitwise exclusive OR operator . . . 123
5.13 Bitwise inclusive OR operator . . . 123
5.14 Logical AND operator . . . 123
5.15 Logical OR operator . . . 124
5.16 Conditional operator . . . 124
5.17 Assignment and compound assignment operators . . . 125
5.18 Comma operator . . . 127
5.19 Constant expressions . . . 127

6 Statements **130**
6.1 Labeled statement . . . 130
6.2 Expression statement . . . 130
6.3 Compound statement or block . . . 130
6.4 Selection statements . . . 131
6.5 Iteration statements . . . 133
6.6 Jump statements . . . 136
6.7 Declaration statement . . . 137
6.8 Ambiguity resolution . . . 138

7 Declarations **140**
7.1 Specifiers . . . 142
7.2 Enumeration declarations . . . 157

© ISO/IEC 2011 – All rights reserved

7.3 Namespaces . . . 161
7.4 The `asm` declaration . . . 173
7.5 Linkage specifications . . . 174
7.6 Attributes . . . 177

8 Declarators **182**
8.1 Type names . . . 183
8.2 Ambiguity resolution . . . 184
8.3 Meaning of declarators . . . 186
8.4 Function definitions . . . 198
8.5 Initializers . . . 202

9 Classes **216**
9.1 Class names . . . 218
9.2 Class members . . . 220
9.3 Member functions . . . 222
9.4 Static members . . . 225
9.5 Unions . . . 227
9.6 Bit-fields . . . 229
9.7 Nested class declarations . . . 229
9.8 Local class declarations . . . 231
9.9 Nested type names . . . 231

10 Derived classes **233**
10.1 Multiple base classes . . . 234
10.2 Member name lookup . . . 236
10.3 Virtual functions . . . 240
10.4 Abstract classes . . . 244

11 Member access control **246**
11.1 Access specifiers . . . 248
11.2 Accessibility of base classes and base class members . . . 249
11.3 Friends . . . 251
11.4 Protected member access . . . 254
11.5 Access to virtual functions . . . 255
11.6 Multiple access . . . 256
11.7 Nested classes . . . 256

12 Special member functions **257**
12.1 Constructors . . . 257
12.2 Temporary objects . . . 260
12.3 Conversions . . . 262
12.4 Destructors . . . 265
12.5 Free store . . . 267
12.6 Initialization . . . 269
12.7 Construction and destruction . . . 275
12.8 Copying and moving class objects . . . 278
12.9 Inheriting constructors . . . 286

13 Overloading **289**
13.1 Overloadable declarations . . . 289

© ISO/IEC 2011 – All rights reserved

13.2 Declaration matching . . . 291
13.3 Overload resolution . . . 292
13.4 Address of overloaded function . . . 311
13.5 Overloaded operators . . . 313
13.6 Built-in operators . . . 317

14 Templates **321**
14.1 Template parameters . . . 322
14.2 Names of template specializations . . . 325
14.3 Template arguments . . . 327
14.4 Type equivalence . . . 333
14.5 Template declarations . . . 334
14.6 Name resolution . . . 352
14.7 Template instantiation and specialization . . . 366
14.8 Function template specializations . . . 378

15 Exception handling **400**
15.1 Throwing an exception . . . 401
15.2 Constructors and destructors . . . 403
15.3 Handling an exception . . . 403
15.4 Exception specifications . . . 405
15.5 Special functions . . . 409

16 Preprocessing directives **411**
16.1 Conditional inclusion . . . 413
16.2 Source file inclusion . . . 414
16.3 Macro replacement . . . 415
16.4 Line control . . . 420
16.5 Error directive . . . 421
16.6 Pragma directive . . . 421
16.7 Null directive . . . 421
16.8 Predefined macro names . . . 421
16.9 Pragma operator . . . 423

17 Library introduction **424**
17.1 General . . . 424
17.2 The C standard library . . . 425
17.3 Definitions . . . 425
17.4 Additional definitions . . . 428
17.5 Method of description (Informative) . . . 428
17.6 Library-wide requirements . . . 434

18 Language support library **454**
18.1 General . . . 454
18.2 Types . . . 454
18.3 Implementation properties . . . 455
18.4 Integer types . . . 464
18.5 Start and termination . . . 465
18.6 Dynamic memory management . . . 467
18.7 Type identification . . . 473
18.8 Exception handling . . . 475

© ISO/IEC 2011 – All rights reserved

18.9 Initializer lists . . . 480
18.10 Other runtime support . . . 481

19 Diagnostics library **484**
19.1 General . . . 484
19.2 Exception classes . . . 484
19.3 Assertions . . . 488
19.4 Error numbers . . . 489
19.5 System error support . . . 489

20 General utilities library **500**
20.1 General . . . 500
20.2 Utility components . . . 500
20.3 Pairs . . . 504
20.4 Tuples . . . 508
20.5 Class template `bitset` . . . 518
20.6 Memory . . . 525
20.7 Smart pointers . . . 540
20.8 Function objects . . . 566
20.9 Metaprogramming and type traits . . . 585
20.10 Compile-time rational arithmetic . . . 602
20.11 Time utilities . . . 605
20.12 Class template `scoped_allocator_adaptor` . . . 620
20.13 Class `type_index` . . . 625

21 Strings library **628**
21.1 General . . . 628
21.2 Character traits . . . 628
21.3 String classes . . . 634
21.4 Class template `basic_string` . . . 638
21.5 Numeric conversions . . . 665
21.6 Hash support . . . 666
21.7 Null-terminated sequence utilities . . . 667

22 Localization library **671**
22.1 General . . . 671
22.2 Header `<locale>` synopsis . . . 671
22.3 Locales . . . 672
22.4 Standard `locale` categories . . . 684
22.5 Standard code conversion facets . . . 725
22.6 C library locales . . . 726

23 Containers library **728**
23.1 General . . . 728
23.2 Container requirements . . . 728
23.3 Sequence containers . . . 754
23.4 Associative containers . . . 786
23.5 Unordered associative containers . . . 803
23.6 Container adaptors . . . 819

24 Iterators library **829**

© ISO/IEC 2011 – All rights reserved

24.1 General . . . 829
24.2 Iterator requirements . . . 829
24.3 Header `<iterator>` synopsis . . . 834
24.4 Iterator primitives . . . 837
24.5 Iterator adaptors . . . 841
24.6 Stream iterators . . . 855

25 Algorithms library **863**
25.1 General . . . 863
25.2 Non-modifying sequence operations . . . 873
25.3 Mutating sequence operations . . . 878
25.4 Sorting and related operations . . . 887
25.5 C library algorithms . . . 900

26 Numerics library **902**
26.1 General . . . 902
26.2 Numeric type requirements . . . 902
26.3 The floating-point environment . . . 903
26.4 Complex numbers . . . 904
26.5 Random number generation . . . 914
26.6 Numeric arrays . . . 959
26.7 Generalized numeric operations . . . 981
26.8 C library . . . 984

27 Input/output library **989**
27.1 General . . . 989
27.2 Iostreams requirements . . . 990
27.3 Forward declarations . . . 990
27.4 Standard iostream objects . . . 992
27.5 Iostreams base classes . . . 994
27.6 Stream buffers . . . 1013
27.7 Formatting and manipulators . . . 1023
27.8 String-based streams . . . 1049
27.9 File-based streams . . . 1061

28 Regular expressions library **1076**
28.1 General . . . 1076
28.2 Definitions . . . 1076
28.3 Requirements . . . 1077
28.4 Header `<regex>` synopsis . . . 1079
28.5 Namespace `std::regex_constants` . . . 1086
28.6 Class `regex_error` . . . 1089
28.7 Class template `regex_traits` . . . 1089
28.8 Class template `basic_regex` . . . 1092
28.9 Class template `sub_match` . . . 1097
28.10 Class template `match_results` . . . 1103
28.11 Regular expression algorithms . . . 1108
28.12 Regular expression iterators . . . 1113
28.13 Modified ECMAScript regular expression grammar . . . 1119

29 Atomic operations library **1122**

© ISO/IEC 2011 – All rights reserved

29.1 General . . . 1122
29.2 Header `<atomic>` synopsis . . . 1122
29.3 Order and consistency . . . 1125
29.4 Lock-free property . . . 1128
29.5 Atomic types . . . 1128
29.6 Operations on atomic types . . . 1132
29.7 Flag type and operations . . . 1137
29.8 Fences . . . 1138

30 Thread support library **1140**
30.1 General . . . 1140
30.2 Requirements . . . 1140
30.3 Threads . . . 1143
30.4 Mutual exclusion . . . 1149
30.5 Condition variables . . . 1162
30.6 Futures . . . 1170

A Grammar summary **1187**
A.1 Keywords . . . 1187
A.2 Lexical conventions . . . 1187
A.3 Basic concepts . . . 1192
A.4 Expressions . . . 1192
A.5 Statements . . . 1195
A.6 Declarations . . . 1196
A.7 Declarators . . . 1200
A.8 Classes . . . 1202
A.9 Derived classes . . . 1203
A.10 Special member functions . . . 1203
A.11 Overloading . . . 1204
A.12 Templates . . . 1204
A.13 Exception handling . . . 1205
A.14 Preprocessing directives . . . 1205

B Implementation quantities **1207**

C Compatibility **1209**
C.1 C++ and ISO C . . . 1209
C.2 C++ and ISO C++ 2003 . . . 1218
C.3 C standard library . . . 1225

D Compatibility features **1229**
D.1 Increment operator with `bool` operand . . . 1229
D.2 `register` keyword . . . 1229
D.3 Implicit declaration of copy functions . . . 1229
D.4 Dynamic exception specifications . . . 1229
D.5 C standard library headers . . . 1229
D.6 Old iostreams members . . . 1230
D.7 `char*` streams . . . 1231
D.8 Function objects . . . 1240
D.9 Binders . . . 1243
D.10 `auto_ptr` . . . 1245

© ISO/IEC 2011 – All rights reserved

D.11 Violating *exception-specifications* . . . 1247

E Universal character names for identifier characters 1249
E.1 Ranges of characters allowed . . . 1249
E.2 Ranges of characters disallowed initially . . . 1249

F Cross references 1250

Index 1268

Index of grammar productions 1297

Index of library names 1300

Index of implementation-defined behavior 1336

© ISO/IEC 2011 – All rights reserved

List of Tables

1 Trigraph sequences 19
2 Alternative tokens 21
3 Identifiers with special meaning 23
4 Keywords 23
5 Alternative representations 24
6 Types of integer constants 25
7 Escape sequences 27
8 String literal concatenations 30

9 Relations on `const` and `volatile` 78

10 *simple-type-specifier*s and the types they specify 154

11 Relationship between operator and function call notation 297
12 Conversions 305

13 Library categories 424
14 C++ library headers 435
15 C++ headers for C library facilities 435
16 C++ headers for freestanding implementations 436
17 `EqualityComparable` requirements 437
18 `LessThanComparable` requirements 437
19 `DefaultConstructible` requirements 437
20 `MoveConstructible` requirements 438
21 `CopyConstructible` requirements (in addition to MoveConstructible) 438
22 `MoveAssignable` requirements 438
23 `CopyAssignable` requirements(in addition to MoveAssignable) 438
24 `Destructible` requirements 438
25 `NullablePointer` requirements 440
26 `Hash` requirements 441
27 Descriptive variable definitions 441
28 Allocator requirements 442

29 Language support library summary 454
30 Header `<cstddef>` synopsis 454
31 Header `<climits>` synopsis 464
32 Header `<cfloat>` synopsis 464
33 Header `<cstdlib>` synopsis 466
34 Header `<csetjmp>` synopsis 482
35 Header `<csignal>` synopsis 482
36 Header `<cstdalign>` synopsis 482
37 Header `<cstdarg>` synopsis 482
38 Header `<cstdbool>` synopsis 482
39 Header `<cstdlib>` synopsis 482
40 Header `<ctime>` synopsis 483

© ISO/IEC 2011 – All rights reserved

41 Diagnostics library summary 484
42 Header <cassert> synopsis 488
43 Header <cerrno> synopsis 489

44 General utilities library summary 500
45 Header <cstdlib> synopsis 539
46 Header <cstring> synopsis 540
47 Primary type category predicates 589
48 Composite type category predicates 589
49 Type property predicates 590
50 Type property queries 595
51 Type relationship predicates 596
52 Const-volatile modifications 597
53 Reference modifications 598
54 Sign modifications 598
55 Array modifications 599
56 Pointer modifications 599
57 Other transformations 600
58 Expressions used to perform ratio arithmetic 604
59 Clock requirements 608
60 Header <ctime> synopsis 619

61 Strings library summary 628
62 Character traits requirements 629
63 basic_string(const Allocator&) effects 643
64 basic_string(const basic_string&) effects 643
65 basic_string(const basic_string&, size_type, size_type, const Allocator&) effects . 643
66 basic_string(const charT*, size_type, const Allocator&) effects 644
67 basic_string(const charT*, const Allocator&) effects 644
68 basic_string(size_t, charT, const Allocator&) effects 644
69 basic_string(const basic_string&, const Allocator&) and basic_string(basic_string&&, const Allocator&) effects 645
70 operator=(const basic_string<charT, traits, Allocator>&) effects 645
71 operator=(const basic_string<charT, traits, Allocator>&&) effects 645
72 compare() results 659
73 Potential mbstate_t data races 668
74 Header <cctype> synopsis 668
75 Header <cwctype> synopsis 669
76 Header <cstring> synopsis 669
77 Header <cwchar> synopsis 669
78 Header <cstdlib> synopsis 669
79 Header <cuchar> synopsis 670

80 Localization library summary 671
81 Locale category facets 675
82 Required specializations 676
83 do_in/do_out result values 694
84 do_unshift result values 694
85 Integer conversions 698
86 Length modifier 698
87 Integer conversions 702

List of Tables

© ISO/IEC 2011 – All rights reserved

88 Floating-point conversions . . . 703
89 Length modifier . . . 703
90 Numeric conversions . . . 703
91 Fill padding . . . 704
92 do_get_date effects . . . 711
93 Header <clocale> synopsis . . . 726
94 Potential setlocale data races . . . 727

95 Containers library summary . . . 728
96 Container requirements . . . 729
97 Reversible container requirements . . . 731
98 Optional container operations . . . 732
99 Allocator-aware container requirements . . . 733
100 Sequence container requirements (in addition to container) . . . 735
101 Optional sequence container operations . . . 737
102 Associative container requirements (in addition to container) . . . 740
103 Unordered associative container requirements (in addition to container) . . . 746

104 Iterators library summary . . . 829
105 Relations among iterator categories . . . 829
106 Iterator requirements . . . 831
107 Input iterator requirements (in addition to Iterator) . . . 831
108 Output iterator requirements (in addition to Iterator) . . . 832
109 Forward iterator requirements (in addition to input iterator) . . . 833
110 Bidirectional iterator requirements (in addition to forward iterator) . . . 833
111 Random access iterator requirements (in addition to bidirectional iterator) . . . 834

112 Algorithms library summary . . . 863
113 Header <cstdlib> synopsis . . . 900

114 Numerics library summary . . . 902
115 Seed sequence requirements . . . 915
116 Uniform random number generator requirements . . . 916
117 Random number engine requirements . . . 917
118 Random number distribution requirements . . . 921
119 Header <cmath> synopsis . . . 984
120 Header <cstdlib> synopsis . . . 985

121 Input/output library summary . . . 989
122 fmtflags effects . . . 999
123 fmtflags constants . . . 999
124 iostate effects . . . 999
125 openmode effects . . . 1000
126 seekdir effects . . . 1000
127 Position type requirements . . . 1004
128 basic_ios::init() effects . . . 1007
129 basic_ios::copyfmt() effects . . . 1008
130 seekoff positioning . . . 1054
131 newoff values . . . 1054
132 File open modes . . . 1064
133 seekoff effects . . . 1067

© ISO/IEC 2011 – All rights reserved

134 Header <cstdio> synopsis . . . 1074
135 Header <cinttypes> synopsis . . . 1075

136 Regular expressions library summary . . . 1076
137 Regular expression traits class requirements . . . 1077
138 syntax_option_type effects . . . 1087
139 regex_constants::match_flag_type effects when obtaining a match against a character container sequence [first,last). . . . 1087
140 error_type values in the C locale . . . 1088
141 match_results assignment operator effects . . . 1105
142 Effects of regex_match algorithm . . . 1109
143 Effects of regex_search algorithm . . . 1110

144 Atomics library summary . . . 1122
145 atomic integral typedefs . . . 1131
146 atomic <inttypes.h> typedefs . . . 1132
147 Atomic arithmetic computations . . . 1136

148 Thread support library summary . . . 1140

149 Standard macros . . . 1225
150 Standard values . . . 1225
151 Standard types . . . 1226
152 Standard structs . . . 1226
153 Standard functions . . . 1226

154 C headers . . . 1229
155 strstreambuf(streamsize) effects . . . 1233
156 strstreambuf(void* (*)(size_t), void (*)(void*)) effects . . . 1233
157 strstreambuf(charT*, streamsize, charT*) effects . . . 1234
158 seekoff positioning . . . 1236
159 newoff values . . . 1236

© ISO/IEC 2011 – All rights reserved

List of Figures

1 Expression category taxonomy . . . 78

2 Directed acyclic graph . . . 234
3 Non-virtual base . . . 235
4 Virtual base . . . 236
5 Virtual and non-virtual base . . . 236
6 Name lookup . . . 239

7 Stream position, offset, and size types [non-normative] . . . 989

© ISO/IEC 2011 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14882 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This third edition cancels and replaces the second edition (ISO/IEC 14882:2003), which has been technically revised.

© ISO/IEC 2011 – All rights reserved

1 General [intro]

1.1 Scope [intro.scope]

1 This International Standard specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, and so this International Standard also defines C++. Other requirements and relaxations of the first requirement appear at various places within this International Standard.

2 C++ is a general purpose programming language based on the C programming language as specified in ISO/IEC 9899:1999*, Programming languages — C* (hereinafter referred to as the *C standard*). In addition to the facilities provided by C, C++ provides additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

1.2 Normative references [intro.refs]

1 The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- Ecma International, *ECMAScript Language Specification*, Standard Ecma-262, third edition, 1999.
- ISO/IEC 2382 (all parts), *Information technology — Vocabulary*
- ISO/IEC 9899:1999, *Programming languages — C*
- ISO/IEC 9899:1999/Cor.1:2001(E), *Programming languages — C, Technical Corrigendum 1*
- ISO/IEC 9899:1999/Cor.2:2004(E), *Programming languages — C, Technical Corrigendum 2*
- ISO/IEC 9899:1999/Cor.3:2007(E), *Programming languages — C, Technical Corrigendum 3*
- ISO/IEC 9945:2003, *Information technology — Portable Operating System Interface (POSIX)*
- ISO/IEC 10646-1:1993, *Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane*
- ISO/IEC TR 19769:2004, *Information technology — Programming languages, their environments and system software interfaces — Extensions for the programming language C to support new character data types*

2 The library described in Clause 7 of ISO/IEC 9899:1999 and Clause 7 of ISO/IEC 9899:1999/Cor.1:2001 and Clause 7 of ISO/IEC 9899:1999/Cor.2:2004 is hereinafter called the *C standard library*.[1]

3 The library described in ISO/IEC TR 19769:2004 is hereinafter called the *C Unicode TR*.

4 The operating system interface described in ISO/IEC 9945:2003 is hereinafter called *POSIX*.

5 The ECMAScript Language Specification described in Standard Ecma-262 is hereinafter called *ECMA-262*.

1) With the qualifications noted in Clauses 18 through 30 and in C.3, the C standard library is a subset of the C++ standard library.

© ISO/IEC 2011 – All rights reserved

1.3 Terms and definitions [intro.defs]

1 For the purposes of this document, the following definitions apply.

2 17.3 defines additional terms that are used only in Clauses 17 through 30 and Annex D.

3 Terms that are used only in a small portion of this International Standard are defined where they are used and italicized where they are defined.

1.3.1 [defns.argument]

argument
actual argument
actual parameter
<function call expression> expression in the comma-separated list bounded by the parentheses

1.3.2 [defns.argument.macro]

argument
actual argument
actual parameter
<function-like macro> sequence of preprocessing tokens in the comma-separated list bounded by the parentheses

1.3.3 [defns.argument.throw]

argument
actual argument
actual parameter
<throw expression> the operand of `throw`

1.3.4 [defns.argument.templ]

argument
actual argument
actual parameter
<template instantiation> expression, *type-id* or *template-name* in the comma-separated list bounded by the angle brackets

1.3.5 [defns.cond.supp]

conditionally-supported
program construct that an implementation is not required to support
[*Note:* Each implementation documents all conditionally-supported constructs that it does not support. — *end note*]

1.3.6 [defns.diagnostic]

diagnostic message
message belonging to an implementation-defined subset of the implementation's output messages

1.3.7 [defns.dynamic.type]

dynamic type
<glvalue> type of the most derived object (1.8) to which the glvalue denoted by a glvalue expression refers

© ISO/IEC 2011 – All rights reserved

[*Example:* if a pointer (8.3.1) `p` whose static type is “pointer to class `B`” is pointing to an object of class `D`, derived from `B` (Clause 10), the dynamic type of the expression `*p` is “`D`.” References (8.3.2) are treated similarly. — *end example*]

1.3.8 **[defns.dynamic.type.prvalue]**
dynamic type
<prvalue> static type of the prvalue expression

1.3.9 **[defns.ill.formed]**
ill-formed program
program that is not well formed

1.3.10 **[defns.impl.defined]**
implementation-defined behavior
behavior, for a well-formed program construct and correct data, that depends on the implementation and that each implementation documents

1.3.11 **[defns.impl.limits]**
implementation limits
restrictions imposed upon programs by the implementation

1.3.12 **[defns.locale.specific]**
locale-specific behavior
behavior that depends on local conventions of nationality, culture, and language that each implementation documents

1.3.13 **[defns.multibyte]**
multibyte character
sequence of one or more bytes representing a member of the extended character set of either the source or the execution environment
[*Note:* The extended character set is a superset of the basic character set (2.3). — *end note*]

1.3.14 **[defns.parameter]**
parameter
formal argument
formal parameter
<function or catch clause> object or reference declared as part of a function declaration or definition or in the catch clause of an exception handler that acquires a value on entry to the function or handler

1.3.15 **[defns.parameter.macro]**
parameter
formal argument
formal parameter
<function-like macro> identifier from the comma-separated list bounded by the parentheses immediately following the macro name

© ISO/IEC 2011 – All rights reserved

1.3.16 **[defns.parameter.templ]**
parameter
formal argument
formal parameter
<template> *template-parameter*

1.3.17 **[defns.signature]**
signature
<function> name, parameter type list (8.3.5), and enclosing namespace (if any)
[*Note:* Signatures are used as a basis for name mangling and linking. — *end note*]

1.3.18 **[defns.signature.templ]**
signature
<function template> name, parameter type list (8.3.5), enclosing namespace (if any), return type, and template parameter list

1.3.19 **[defns.signature.spec]**
signature
<function template specialization> signature of the template of which it is a specialization and its template arguments (whether explicitly specified or deduced)

1.3.20 **[defns.signature.member]**
signature
<class member function> name, parameter type list (8.3.5), class of which the function is a member, *cv*-qualifiers (if any), and *ref-qualifier* (if any)

1.3.21 **[defns.signature.member.templ]**
signature
<class member function template> name, parameter type list (8.3.5), class of which the function is a member, *cv*-qualifiers (if any), *ref-qualifier* (if any), return type, and template parameter list

1.3.22 **[defns.signature.member.spec]**
signature
<class member function template specialization> signature of the member function template of which it is a specialization and its template arguments (whether explicitly specified or deduced)

1.3.23 **[defns.static.type]**
static type
type of an expression (3.9) resulting from analysis of the program without considering execution semantics
[*Note:* The static type of an expression depends only on the form of the program in which the expression appears, and does not change while the program is executing. — *end note*]

1.3.24 **[defns.undefined]**
undefined behavior
behavior for which this International Standard imposes no requirements
[*Note:* Undefined behavior may be expected when this International Standard omits any explicit definition of

© ISO/IEC 2011 – All rights reserved