INTERNATIONAL STANDARD

ISO/IEC 14882

Fourth edition
2014-12-15

# Information technology — Programming languages — C++

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

Reference number
ISO/IEC 14882:2014(E)

# Contents

**Contents** iii

**List of Tables** xi

**List of Figures** xv

**Foreword** xvi

**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 . . . 5
1.6 Syntax notation . . . 6
1.7 The C++ memory model . . . 6
1.8 The C++ object model . . . 7
1.9 Program execution . . . 8
1.10 Multi-threaded executions and data races . . . 11
1.11 Acknowledgments . . . 15

**2 Lexical conventions** 16
2.1 Separate translation . . . 16
2.2 Phases of translation . . . 16
2.3 Character sets . . . 17
2.4 Trigraph sequences . . . 18
2.5 Preprocessing tokens . . . 19
2.6 Alternative tokens . . . 20
2.7 Tokens . . . 20
2.8 Comments . . . 20
2.9 Header names . . . 20
2.10 Preprocessing numbers . . . 21
2.11 Identifiers . . . 21
2.12 Keywords . . . 22
2.13 Operators and punctuators . . . 22
2.14 Literals . . . 23

**3 Basic concepts** 32
3.1 Declarations and definitions . . . 32
3.2 One definition rule . . . 34
3.3 Scope . . . 37
3.4 Name lookup . . . 42
3.5 Program and linkage . . . 56
3.6 Start and termination . . . 59
3.7 Storage duration . . . 62
3.8 Object lifetime . . . 66

3.9 Types . . . 69
3.10 Lvalues and rvalues . . . 75
3.11 Alignment . . . 76

**4 Standard conversions** **78**
4.1 Lvalue-to-rvalue conversion . . . 79
4.2 Array-to-pointer conversion . . . 79
4.3 Function-to-pointer conversion . . . 79
4.4 Qualification conversions . . . 80
4.5 Integral promotions . . . 81
4.6 Floating point promotion . . . 81
4.7 Integral conversions . . . 81
4.8 Floating point conversions . . . 82
4.9 Floating-integral conversions . . . 82
4.10 Pointer conversions . . . 82
4.11 Pointer to member conversions . . . 82
4.12 Boolean conversions . . . 83
4.13 Integer conversion rank . . . 83

**5 Expressions** **84**
5.1 Primary expressions . . . 87
5.2 Postfix expressions . . . 97
5.3 Unary expressions . . . 108
5.4 Explicit type conversion (cast notation) . . . 117
5.5 Pointer-to-member operators . . . 118
5.6 Multiplicative operators . . . 118
5.7 Additive operators . . . 119
5.8 Shift operators . . . 120
5.9 Relational operators . . . 120
5.10 Equality operators . . . 121
5.11 Bitwise AND operator . . . 122
5.12 Bitwise exclusive OR operator . . . 122
5.13 Bitwise inclusive OR operator . . . 122
5.14 Logical AND operator . . . 123
5.15 Logical OR operator . . . 123
5.16 Conditional operator . . . 123
5.17 Assignment and compound assignment operators . . . 125
5.18 Comma operator . . . 126
5.19 Constant expressions . . . 126

**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 . . . 132
6.6 Jump statements . . . 135
6.7 Declaration statement . . . 136
6.8 Ambiguity resolution . . . 137

**7 Declarations** **139**

7.1 Specifiers . . . 140
7.2 Enumeration declarations . . . 157
7.3 Namespaces . . . 161
7.4 The `asm` declaration . . . 173
7.5 Linkage specifications . . . 173
7.6 Attributes . . . 176

**8 Declarators** **181**
8.1 Type names . . . 182
8.2 Ambiguity resolution . . . 183
8.3 Meaning of declarators . . . 184
8.4 Function definitions . . . 196
8.5 Initializers . . . 199

**9 Classes** **214**
9.1 Class names . . . 216
9.2 Class members . . . 218
9.3 Member functions . . . 220
9.4 Static members . . . 223
9.5 Unions . . . 224
9.6 Bit-fields . . . 226
9.7 Nested class declarations . . . 227
9.8 Local class declarations . . . 228
9.9 Nested type names . . . 228

**10 Derived classes** **230**
10.1 Multiple base classes . . . 231
10.2 Member name lookup . . . 233
10.3 Virtual functions . . . 236
10.4 Abstract classes . . . 240

**11 Member access control** **242**
11.1 Access specifiers . . . 243
11.2 Accessibility of base classes and base class members . . . 244
11.3 Friends . . . 247
11.4 Protected member access . . . 250
11.5 Access to virtual functions . . . 251
11.6 Multiple access . . . 251
11.7 Nested classes . . . 251

**12 Special member functions** **253**
12.1 Constructors . . . 253
12.2 Temporary objects . . . 255
12.3 Conversions . . . 258
12.4 Destructors . . . 260
12.5 Free store . . . 263
12.6 Initialization . . . 265
12.7 Construction and destruction . . . 270
12.8 Copying and moving class objects . . . 273
12.9 Inheriting constructors . . . 280

**13 Overloading** **284**
13.1 Overloadable declarations . . . 284
13.2 Declaration matching . . . 286
13.3 Overload resolution . . . 287
13.4 Address of overloaded function . . . 307
13.5 Overloaded operators . . . 308
13.6 Built-in operators . . . 312

**14 Templates** **316**
14.1 Template parameters . . . 317
14.2 Names of template specializations . . . 320
14.3 Template arguments . . . 322
14.4 Type equivalence . . . 328
14.5 Template declarations . . . 329
14.6 Name resolution . . . 346
14.7 Template instantiation and specialization . . . 359
14.8 Function template specializations . . . 371

**15 Exception handling** **392**
15.1 Throwing an exception . . . 393
15.2 Constructors and destructors . . . 395
15.3 Handling an exception . . . 395
15.4 Exception specifications . . . 397
15.5 Special functions . . . 400

**16 Preprocessing directives** **403**
16.1 Conditional inclusion . . . 404
16.2 Source file inclusion . . . 405
16.3 Macro replacement . . . 406
16.4 Line control . . . 411
16.5 Error directive . . . 412
16.6 Pragma directive . . . 412
16.7 Null directive . . . 412
16.8 Predefined macro names . . . 412
16.9 Pragma operator . . . 413

**17 Library introduction** **414**
17.1 General . . . 414
17.2 The C standard library . . . 415
17.3 Definitions . . . 415
17.4 Additional definitions . . . 418
17.5 Method of description (Informative) . . . 418
17.6 Library-wide requirements . . . 423

**18 Language support library** **443**
18.1 General . . . 443
18.2 Types . . . 443
18.3 Implementation properties . . . 444
18.4 Integer types . . . 453
18.5 Start and termination . . . 455
18.6 Dynamic memory management . . . 456

18.7 Type identification . . . 463
18.8 Exception handling . . . 465
18.9 Initializer lists . . . 470
18.10 Other runtime support . . . 471

**19 Diagnostics library** **474**
19.1 General . . . 474
19.2 Exception classes . . . 474
19.3 Assertions . . . 478
19.4 Error numbers . . . 478
19.5 System error support . . . 478

**20 General utilities library** **490**
20.1 General . . . 490
20.2 Utility components . . . 490
20.3 Pairs . . . 495
20.4 Tuples . . . 500
20.5 Compile-time integer sequences . . . 510
20.6 Class template `bitset` . . . 511
20.7 Memory . . . 519
20.8 Smart pointers . . . 534
20.9 Function objects . . . 562
20.10 Metaprogramming and type traits . . . 584
20.11 Compile-time rational arithmetic . . . 603
20.12 Time utilities . . . 606
20.13 Class template `scoped_allocator_adaptor` . . . 622
20.14 Class `type_index` . . . 629

**21 Strings library** **631**
21.1 General . . . 631
21.2 Character traits . . . 631
21.3 String classes . . . 637
21.4 Class template `basic_string` . . . 641
21.5 Numeric conversions . . . 669
21.6 Hash support . . . 671
21.7 Suffix for `basic_string` literals . . . 671
21.8 Null-terminated sequence utilities . . . 671

**22 Localization library** **675**
22.1 General . . . 675
22.2 Header `<locale>` synopsis . . . 675
22.3 Locales . . . 676
22.4 Standard `locale` categories . . . 689
22.5 Standard code conversion facets . . . 729
22.6 C library locales . . . 731

**23 Containers library** **732**
23.1 General . . . 732
23.2 Container requirements . . . 732
23.3 Sequence containers . . . 760
23.4 Associative containers . . . 791

23.5 Unordered associative containers . . . 808
23.6 Container adaptors . . . 825

**24 Iterators library** **835**
24.1 General . . . 835
24.2 Iterator requirements . . . 835
24.3 Header `<iterator>` synopsis . . . 840
24.4 Iterator primitives . . . 843
24.5 Iterator adaptors . . . 847
24.6 Stream iterators . . . 860
24.7 range access . . . 867

**25 Algorithms library** **869**
25.1 General . . . 869
25.2 Non-modifying sequence operations . . . 880
25.3 Mutating sequence operations . . . 885
25.4 Sorting and related operations . . . 893
25.5 C library algorithms . . . 906

**26 Numerics library** **908**
26.1 General . . . 908
26.2 Numeric type requirements . . . 908
26.3 The floating-point environment . . . 909
26.4 Complex numbers . . . 910
26.5 Random number generation . . . 921
26.6 Numeric arrays . . . 966
26.7 Generalized numeric operations . . . 987
26.8 C library . . . 990

**27 Input/output library** **995**
27.1 General . . . 995
27.2 Iostreams requirements . . . 995
27.3 Forward declarations . . . 996
27.4 Standard iostream objects . . . 998
27.5 Iostreams base classes . . . 1000
27.6 Stream buffers . . . 1019
27.7 Formatting and manipulators . . . 1029
27.8 String-based streams . . . 1058
27.9 File-based streams . . . 1069

**28 Regular expressions library** **1085**
28.1 General . . . 1085
28.2 Definitions . . . 1085
28.3 Requirements . . . 1086
28.4 Header `<regex>` synopsis . . . 1088
28.5 Namespace `std::regex_constants` . . . 1095
28.6 Class `regex_error` . . . 1098
28.7 Class template `regex_traits` . . . 1098
28.8 Class template `basic_regex` . . . 1101
28.9 Class template `sub_match` . . . 1108
28.10 Class template `match_results` . . . 1114

28.11 Regular expression algorithms . . . 1120
28.12 Regular expression iterators . . . 1125
28.13 Modified ECMAScript regular expression grammar . . . 1131

**29 Atomic operations library** **1134**
29.1 General . . . 1134
29.2 Header `<atomic>` synopsis . . . 1134
29.3 Order and consistency . . . 1137
29.4 Lock-free property . . . 1139
29.5 Atomic types . . . 1139
29.6 Operations on atomic types . . . 1143
29.7 Flag type and operations . . . 1149
29.8 Fences . . . 1150

**30 Thread support library** **1151**
30.1 General . . . 1151
30.2 Requirements . . . 1151
30.3 Threads . . . 1154
30.4 Mutual exclusion . . . 1159
30.5 Condition variables . . . 1179
30.6 Futures . . . 1187

**A Grammar summary** **1205**
A.1 Keywords . . . 1205
A.2 Lexical conventions . . . 1205
A.3 Basic concepts . . . 1209
A.4 Expressions . . . 1210
A.5 Statements . . . 1213
A.6 Declarations . . . 1214
A.7 Declarators . . . 1218
A.8 Classes . . . 1220
A.9 Derived classes . . . 1220
A.10 Special member functions . . . 1221
A.11 Overloading . . . 1221
A.12 Templates . . . 1222
A.13 Exception handling . . . 1222
A.14 Preprocessing directives . . . 1223

**B Implementation quantities** **1225**

**C Compatibility** **1227**
C.1 C++ and ISO C . . . 1227
C.2 C++ and ISO C++ 2003 . . . 1235
C.3 C++ and ISO C++ 2011 . . . 1242
C.4 C standard library . . . 1243

**D Compatibility features** **1247**
D.1 Increment operator with `bool` operand . . . 1247
D.2 `register` keyword . . . 1247
D.3 Implicit declaration of copy functions . . . 1247
D.4 Dynamic exception specifications . . . 1247

D.5 C standard library headers . . . . . 1247
D.6 Old iostreams members . . . . . 1248
D.7 `char*` streams . . . . . 1249
D.8 Function objects . . . . . 1258
D.9 Binders . . . . . 1262
D.10 `auto_ptr` . . . . . 1263
D.11 Violating *exception-specifications* . . . . . 1266
D.12 Random shuffle . . . . . 1266

**E Universal character names for identifier characters** **1268**
E.1 Ranges of characters allowed . . . . . 1268
E.2 Ranges of characters disallowed initially . . . . . 1268

**F Cross references** **1269**

**Index** **1287**

**Index of grammar productions** **1316**

**Index of library names** **1319**

**Index of implementation-defined behavior** **1356**

# List of Tables

1 Trigraph sequences . . . . . 18
2 Alternative tokens . . . . . 20
3 Identifiers with special meaning . . . . . 22
4 Keywords . . . . . 22
5 Alternative representations . . . . . 22
6 Types of integer literals . . . . . 24
7 Escape sequences . . . . . 26
8 String literal concatenations . . . . . 29

9 Relations on `const` and `volatile` . . . . . 74

10 *simple-type-specifier*s and the types they specify . . . . . 151

11 Relationship between operator and function call notation . . . . . 292
12 Conversions . . . . . 300

13 Library categories . . . . . 414
14 C++ library headers . . . . . 424
15 C++ headers for C library facilities . . . . . 424
16 C++ headers for freestanding implementations . . . . . 425
17 `EqualityComparable` requirements . . . . . 426
18 `LessThanComparable` requirements . . . . . 426
19 `DefaultConstructible` requirements . . . . . 427
20 `MoveConstructible` requirements . . . . . 427
21 `CopyConstructible` requirements (in addition to `MoveConstructible`) . . . . . 427
22 `MoveAssignable` requirements . . . . . 427
23 `CopyAssignable` requirements (in addition to `MoveAssignable`) . . . . . 427
24 `Destructible` requirements . . . . . 427
25 `NullablePointer` requirements . . . . . 429
26 `Hash` requirements . . . . . 430
27 Descriptive variable definitions . . . . . 430
28 Allocator requirements . . . . . 431

29 Language support library summary . . . . . 443
30 Header `<cstddef>` synopsis . . . . . 443
31 Header `<climits>` synopsis . . . . . 453
32 Header `<cfloat>` synopsis . . . . . 453
33 Header `<cstdlib>` synopsis . . . . . 455
34 Header `<csetjmp>` synopsis . . . . . 472
35 Header `<csignal>` synopsis . . . . . 472
36 Header `<cstdalign>` synopsis . . . . . 472
37 Header `<cstdarg>` synopsis . . . . . 472
38 Header `<cstdbool>` synopsis . . . . . 473
39 Header `<cstdlib>` synopsis . . . . . 473
40 Header `<ctime>` synopsis . . . . . 473

41 Diagnostics library summary . . . 474
42 Header <cassert> synopsis . . . 478
43 Header <cerrno> synopsis . . . 479

44 General utilities library summary . . . 490
45 Header <cstdlib> synopsis . . . 533
46 Header <cstring> synopsis . . . 534
47 Primary type category predicates . . . 588
48 Composite type category predicates . . . 589
49 Type property predicates . . . 590
50 Type property queries . . . 596
51 Type relationship predicates . . . 597
52 Const-volatile modifications . . . 598
53 Reference modifications . . . 598
54 Sign modifications . . . 599
55 Array modifications . . . 600
56 Pointer modifications . . . 600
57 Other transformations . . . 601
58 Expressions used to perform ratio arithmetic . . . 605
59 Clock requirements . . . 609
60 Header <ctime> synopsis . . . 622

61 Strings library summary . . . 631
62 Character traits requirements . . . 632
63 `basic_string(const Allocator&)` effects . . . 645
64 `basic_string(const basic_string&)` effects . . . 646
65 `basic_string(const basic_string&, size_type, size_type, const Allocator&)` effects . . . 646
66 `basic_string(const charT*, size_type, const Allocator&)` effects . . . 647
67 `basic_string(const charT*, const Allocator&)` effects . . . 647
68 `basic_string(size_t, charT, const Allocator&)` effects . . . 647
69 `basic_string(const basic_string&, const Allocator&)` and `basic_string(basic_string&&, const Allocator&)` effects . . . 648
70 `operator=(const basic_string&)` effects . . . 648
71 `operator=(basic_string&&)` effects . . . 649
72 `compare()` results . . . 663
73 Potential `mbstate_t` data races . . . 673
74 Header <cctype> synopsis . . . 673
75 Header <cwctype> synopsis . . . 673
76 Header <cstring> synopsis . . . 673
77 Header <cwchar> synopsis . . . 674
78 Header <cstdlib> synopsis . . . 674
79 Header <cuchar> synopsis . . . 674

80 Localization library summary . . . 675
81 Locale category facets . . . 679
82 Required specializations . . . 679
83 `do_in/do_out` result values . . . 699
84 `do_unshift` result values . . . 699
85 Integer conversions . . . 703
86 Length modifier . . . 703
87 Integer conversions . . . 707

88 Floating-point conversions . . . 707
89 Length modifier . . . 708
90 Numeric conversions . . . 708
91 Fill padding . . . 709
92 `do_get_date` effects . . . 716
93 Header `<clocale>` synopsis . . . 731
94 Potential `setlocale` data races . . . 731

95 Containers library summary . . . 732
96 Container requirements . . . 733
97 Reversible container requirements . . . 735
98 Optional container operations . . . 736
99 Allocator-aware container requirements . . . 738
100 Sequence container requirements (in addition to container) . . . 740
101 Optional sequence container operations . . . 742
102 Associative container requirements (in addition to container) . . . 744
103 Unordered associative container requirements (in addition to container) . . . 752

104 Iterators library summary . . . 835
105 Relations among iterator categories . . . 835
106 Iterator requirements . . . 836
107 Input iterator requirements (in addition to Iterator) . . . 837
108 Output iterator requirements (in addition to Iterator) . . . 838
109 Forward iterator requirements (in addition to input iterator) . . . 839
110 Bidirectional iterator requirements (in addition to forward iterator) . . . 839
111 Random access iterator requirements (in addition to bidirectional iterator) . . . 840

112 Algorithms library summary . . . 869
113 Header `<cstdlib>` synopsis . . . 906

114 Numerics library summary . . . 908
115 Seed sequence requirements . . . 923
116 Uniform random number generator requirements . . . 924
117 Random number engine requirements . . . 925
118 Random number distribution requirements . . . 928
119 Header `<cmath>` synopsis . . . 991
120 Header `<cstdlib>` synopsis . . . 991

121 Input/output library summary . . . 995
122 `fmtflags` effects . . . 1005
123 `fmtflags` constants . . . 1005
124 `iostate` effects . . . 1005
125 `openmode` effects . . . 1005
126 `seekdir` effects . . . 1006
127 Position type requirements . . . 1010
128 `basic_ios::init()` effects . . . 1012
129 `basic_ios::copyfmt()` effects . . . 1014
130 `seekoff` positioning . . . 1062
131 `newoff` values . . . 1063
132 File open modes . . . 1073
133 `seekoff` effects . . . 1075

134 Header `<cstdio>` synopsis . . . . . 1083
135 Header `<cinttypes>` synopsis . . . . . 1084

136 Regular expressions library summary . . . . . 1085
137 Regular expression traits class requirements . . . . . 1086
138 `syntax_option_type` effects . . . . . 1096
139 `regex_constants::match_flag_type` effects when obtaining a match against a character container sequence `[first,last)`. . . . . . 1096
140 `error_type` values in the C locale . . . . . 1097
141 Character class names and corresponding `ctype` masks . . . . . 1102
142 `match_results` assignment operator effects . . . . . 1117
143 Effects of `regex_match` algorithm . . . . . 1120
144 Effects of `regex_search` algorithm . . . . . 1122

145 Atomics library summary . . . . . 1134
146 `atomic` integral typedefs . . . . . 1143
147 `atomic <inttypes.h>` typedefs . . . . . 1144
148 Atomic arithmetic computations . . . . . 1148

149 Thread support library summary . . . . . 1151

150 Standard macros . . . . . 1243
151 Standard values . . . . . 1243
152 Standard types . . . . . 1244
153 Standard structs . . . . . 1244
154 Standard functions . . . . . 1245

155 C headers . . . . . 1247
156 `strstreambuf(streamsize)` effects . . . . . 1251
157 `strstreambuf(void* (*)(size_t), void (*)(void*))` effects . . . . . 1251
158 `strstreambuf(charT*, streamsize, charT*)` effects . . . . . 1251
159 `seekoff` positioning . . . . . 1254
160 `newoff` values . . . . . 1254

# List of Figures

1 Expression category taxonomy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

2 Directed acyclic graph . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 231
3 Non-virtual base . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232
4 Virtual base . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 233
5 Virtual and non-virtual base . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 233
6 Name lookup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 235

7 Stream position, offset, and size types [non-normative] . . . . . . . . . . . . . . . . . . . . 995

# 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 22, *Programming languages, their environments and system software interfaces*

This fourth edition cancels and replaces the third edition (ISO/IEC 14882:2011), of which it constitutes a minor revision.

# 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 described 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.2:2003 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.4, the C standard library is a subset of the C++ standard library.