

# American National Standard

INCITS/ISO/IEC 14882:2017 (2018)

(ISO/IEC 14882:2017, IDT)

*Programming languages - C++*

**Developed by**



*Where IT all begins*



## INCITS/ISO/IEC 14882:2017 (2018)

### PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

**Adopted by INCITS (InterNational Committee for Information Technology Standards) as an American National Standard.**

Date of ANSI Approval: 6/29/2018

Published by American National Standards Institute,  
25 West 43rd Street, New York, New York 10036

Copyright 2018 by Information Technology Industry Council  
(ITI). All rights reserved.

These materials are subject to copyright claims of International Standardization Organization (ISO), International Electrotechnical Commission (IEC), American National Standards Institute (ANSI), and Information Technology Industry Council (ITI). Not for resale. No part of this publication may be reproduced in any form, including an electronic retrieval system, without the prior written permission of ITI. All requests pertaining to this standard should be submitted to ITI, 1101 K Street NW, Suite 610, Washington DC 20005.

Printed in the United States of America

Fifth edition  
2017-12

---

---

## Programming languages — C++

*Langages de programmation — C++*



Reference number  
ISO/IEC 14882:2017(E)

© ISO/IEC 2017



**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Ch. de Blandonnet 8 • CP 401  
CH-1214 Vernier, Geneva, Switzerland  
Tel. +41 22 749 01 11  
Fax +41 22 749 09 47  
copyright@iso.org  
www.iso.org

# Contents

<b>Foreword</b>	<b>xi</b>
<b>1 Scope</b>	<b>1</b>
<b>2 Normative references</b>	<b>2</b>
<b>3 Terms and definitions</b>	<b>3</b>
<b>4 General principles</b>	<b>7</b>
4.1 Implementation compliance . . . . .	7
4.2 Structure of this document . . . . .	8
4.3 Syntax notation . . . . .	8
4.4 The C++ memory model . . . . .	8
4.5 The C++ object model . . . . .	9
4.6 Program execution . . . . .	11
4.7 Multi-threaded executions and data races . . . . .	15
4.8 Acknowledgments . . . . .	20
<b>5 Lexical conventions</b>	<b>22</b>
5.1 Separate translation . . . . .	22
5.2 Phases of translation . . . . .	22
5.3 Character sets . . . . .	23
5.4 Preprocessing tokens . . . . .	24
5.5 Alternative tokens . . . . .	25
5.6 Tokens . . . . .	25
5.7 Comments . . . . .	26
5.8 Header names . . . . .	26
5.9 Preprocessing numbers . . . . .	26
5.10 Identifiers . . . . .	27
5.11 Keywords . . . . .	28
5.12 Operators and punctuators . . . . .	29
5.13 Literals . . . . .	29
<b>6 Basic concepts</b>	<b>39</b>
6.1 Declarations and definitions . . . . .	39
6.2 One-definition rule . . . . .	41
6.3 Scope . . . . .	44
6.4 Name lookup . . . . .	50
6.5 Program and linkage . . . . .	63
6.6 Start and termination . . . . .	66
6.7 Storage duration . . . . .	70
6.8 Object lifetime . . . . .	74
6.9 Types . . . . .	77
6.10 Lvalues and rvalues . . . . .	83
6.11 Alignment . . . . .	84

<b>7</b>	<b>Standard conversions</b>	<b>86</b>
7.1	Lvalue-to-rvalue conversion . . . . .	87
7.2	Array-to-pointer conversion . . . . .	87
7.3	Function-to-pointer conversion . . . . .	88
7.4	Temporary materialization conversion . . . . .	88
7.5	Qualification conversions . . . . .	88
7.6	Integral promotions . . . . .	89
7.7	Floating-point promotion . . . . .	89
7.8	Integral conversions . . . . .	89
7.9	Floating-point conversions . . . . .	90
7.10	Floating-integral conversions . . . . .	90
7.11	Pointer conversions . . . . .	90
7.12	Pointer to member conversions . . . . .	90
7.13	Function pointer conversions . . . . .	91
7.14	Boolean conversions . . . . .	91
7.15	Integer conversion rank . . . . .	91
<b>8</b>	<b>Expressions</b>	<b>93</b>
8.1	Primary expressions . . . . .	96
8.2	Postfix expressions . . . . .	109
8.3	Unary expressions . . . . .	120
8.4	Explicit type conversion (cast notation) . . . . .	129
8.5	Pointer-to-member operators . . . . .	130
8.6	Multiplicative operators . . . . .	131
8.7	Additive operators . . . . .	131
8.8	Shift operators . . . . .	132
8.9	Relational operators . . . . .	133
8.10	Equality operators . . . . .	133
8.11	Bitwise AND operator . . . . .	135
8.12	Bitwise exclusive OR operator . . . . .	135
8.13	Bitwise inclusive OR operator . . . . .	135
8.14	Logical AND operator . . . . .	135
8.15	Logical OR operator . . . . .	135
8.16	Conditional operator . . . . .	136
8.17	Throwing an exception . . . . .	137
8.18	Assignment and compound assignment operators . . . . .	137
8.19	Comma operator . . . . .	138
8.20	Constant expressions . . . . .	139
<b>9</b>	<b>Statements</b>	<b>144</b>
9.1	Labeled statement . . . . .	145
9.2	Expression statement . . . . .	145
9.3	Compound statement or block . . . . .	145
9.4	Selection statements . . . . .	145
9.5	Iteration statements . . . . .	148
9.6	Jump statements . . . . .	150
9.7	Declaration statement . . . . .	152
9.8	Ambiguity resolution . . . . .	153

<b>10</b>	<b>Declarations</b>	<b>155</b>
10.1	Specifiers . . . . .	157
10.2	Enumeration declarations . . . . .	174
10.3	Namespaces . . . . .	178
10.4	The <code>asm</code> declaration . . . . .	191
10.5	Linkage specifications . . . . .	191
10.6	Attributes . . . . .	194
<b>11</b>	<b>Declarators</b>	<b>201</b>
11.1	Type names . . . . .	202
11.2	Ambiguity resolution . . . . .	203
11.3	Meaning of declarators . . . . .	204
11.4	Function definitions . . . . .	216
11.5	Structured binding declarations . . . . .	219
11.6	Initializers . . . . .	220
<b>12</b>	<b>Classes</b>	<b>237</b>
12.1	Class names . . . . .	239
12.2	Class members . . . . .	241
12.3	Unions . . . . .	251
12.4	Local class declarations . . . . .	254
<b>13</b>	<b>Derived classes</b>	<b>255</b>
13.1	Multiple base classes . . . . .	256
13.2	Member name lookup . . . . .	258
13.3	Virtual functions . . . . .	261
13.4	Abstract classes . . . . .	265
<b>14</b>	<b>Member access control</b>	<b>267</b>
14.1	Access specifiers . . . . .	268
14.2	Accessibility of base classes and base class members . . . . .	269
14.3	Friends . . . . .	272
14.4	Protected member access . . . . .	275
14.5	Access to virtual functions . . . . .	276
14.6	Multiple access . . . . .	276
14.7	Nested classes . . . . .	276
<b>15</b>	<b>Special member functions</b>	<b>278</b>
15.1	Constructors . . . . .	278
15.2	Temporary objects . . . . .	281
15.3	Conversions . . . . .	283
15.4	Destructors . . . . .	286
15.5	Free store . . . . .	289
15.6	Initialization . . . . .	291
15.7	Construction and destruction . . . . .	298
15.8	Copying and moving class objects . . . . .	301
<b>16</b>	<b>Overloading</b>	<b>309</b>
16.1	Overloadable declarations . . . . .	309
16.2	Declaration matching . . . . .	311
16.3	Overload resolution . . . . .	312

16.4	Address of overloaded function . . . . .	333
16.5	Overloaded operators . . . . .	334
16.6	Built-in operators . . . . .	339
<b>17</b>	<b>Templates</b>	<b>342</b>
17.1	Template parameters . . . . .	343
17.2	Names of template specializations . . . . .	347
17.3	Template arguments . . . . .	348
17.4	Type equivalence . . . . .	354
17.5	Template declarations . . . . .	355
17.6	Name resolution . . . . .	373
17.7	Template instantiation and specialization . . . . .	388
17.8	Function template specializations . . . . .	400
17.9	Deduction guides . . . . .	421
<b>18</b>	<b>Exception handling</b>	<b>422</b>
18.1	Throwing an exception . . . . .	423
18.2	Constructors and destructors . . . . .	425
18.3	Handling an exception . . . . .	425
18.4	Exception specifications . . . . .	427
18.5	Special functions . . . . .	430
<b>19</b>	<b>Preprocessing directives</b>	<b>432</b>
19.1	Conditional inclusion . . . . .	433
19.2	Source file inclusion . . . . .	435
19.3	Macro replacement . . . . .	436
19.4	Line control . . . . .	441
19.5	Error directive . . . . .	442
19.6	Pragma directive . . . . .	442
19.7	Null directive . . . . .	442
19.8	Predefined macro names . . . . .	442
19.9	Pragma operator . . . . .	444
<b>20</b>	<b>Library introduction</b>	<b>445</b>
20.1	General . . . . .	445
20.2	The C standard library . . . . .	446
20.3	Definitions . . . . .	446
20.4	Method of description (Informative) . . . . .	449
20.5	Library-wide requirements . . . . .	454
<b>21</b>	<b>Language support library</b>	<b>476</b>
21.1	General . . . . .	476
21.2	Common definitions . . . . .	476
21.3	Implementation properties . . . . .	481
21.4	Integer types . . . . .	490
21.5	Start and termination . . . . .	491
21.6	Dynamic memory management . . . . .	492
21.7	Type identification . . . . .	500
21.8	Exception handling . . . . .	502
21.9	Initializer lists . . . . .	507
21.10	Other runtime support . . . . .	508



<b>22</b>	<b>Diagnostics library</b>	<b>511</b>
22.1	General . . . . .	511
22.2	Exception classes . . . . .	511
22.3	Assertions . . . . .	515
22.4	Error numbers . . . . .	515
22.5	System error support . . . . .	517
<b>23</b>	<b>General utilities library</b>	<b>528</b>
23.1	General . . . . .	528
23.2	Utility components . . . . .	528
23.3	Compile-time integer sequences . . . . .	536
23.4	Pairs . . . . .	537
23.5	Tuples . . . . .	541
23.6	Optional objects . . . . .	553
23.7	Variants . . . . .	567
23.8	Storage for any type . . . . .	580
23.9	Bitsets . . . . .	586
23.10	Memory . . . . .	592
23.11	Smart pointers . . . . .	607
23.12	Memory resources . . . . .	634
23.13	Class template <code>scoped_allocator_adaptor</code> . . . . .	645
23.14	Function objects . . . . .	651
23.15	Metaprogramming and type traits . . . . .	675
23.16	Compile-time rational arithmetic . . . . .	699
23.17	Time utilities . . . . .	702
23.18	Class <code>type_index</code> . . . . .	719
23.19	Execution policies . . . . .	720
<b>24</b>	<b>Strings library</b>	<b>723</b>
24.1	General . . . . .	723
24.2	Character traits . . . . .	723
24.3	String classes . . . . .	729
24.4	String view classes . . . . .	762
24.5	Null-terminated sequence utilities . . . . .	772
<b>25</b>	<b>Localization library</b>	<b>778</b>
25.1	General . . . . .	778
25.2	Header <code>&lt;locale&gt;</code> synopsis . . . . .	778
25.3	Locales . . . . .	780
25.4	Standard <code>locale</code> categories . . . . .	787
25.5	C library locales . . . . .	825
<b>26</b>	<b>Containers library</b>	<b>826</b>
26.1	General . . . . .	826
26.2	Container requirements . . . . .	826
26.3	Sequence containers . . . . .	864
26.4	Associative containers . . . . .	896
26.5	Unordered associative containers . . . . .	918
26.6	Container adaptors . . . . .	942

<b>27 Iterators library</b>	<b>952</b>
27.1 General	952
27.2 Iterator requirements	952
27.3 Header <code>&lt;iterator&gt;</code> synopsis	958
27.4 Iterator primitives	961
27.5 Iterator adaptors	964
27.6 Stream iterators	977
27.7 Range access	984
27.8 Container access	985
<b>28 Algorithms library</b>	<b>986</b>
28.1 General	986
28.2 Header <code>&lt;algorithm&gt;</code> synopsis	986
28.3 Algorithms requirements	1005
28.4 Parallel algorithms	1006
28.5 Non-modifying sequence operations	1009
28.6 Mutating sequence operations	1017
28.7 Sorting and related operations	1027
28.8 C library algorithms	1046
<b>29 Numerics library</b>	<b>1047</b>
29.1 General	1047
29.2 Definitions	1047
29.3 Numeric type requirements	1047
29.4 The floating-point environment	1048
29.5 Complex numbers	1049
29.6 Random number generation	1059
29.7 Numeric arrays	1102
29.8 Generalized numeric operations	1122
29.9 Mathematical functions for floating-point types	1136
<b>30 Input/output library</b>	<b>1153</b>
30.1 General	1153
30.2 Iostreams requirements	1154
30.3 Forward declarations	1154
30.4 Standard iostream objects	1156
30.5 Iostreams base classes	1158
30.6 Stream buffers	1175
30.7 Formatting and manipulators	1184
30.8 String-based streams	1211
30.9 File-based streams	1221
30.10 File systems	1235
30.11 C library files	1288
<b>31 Regular expressions library</b>	<b>1292</b>
31.1 General	1292
31.2 Definitions	1292
31.3 Requirements	1293
31.4 Header <code>&lt;regex&gt;</code> synopsis	1295
31.5 Namespace <code>std::regex_constants</code>	1301
31.6 Class <code>regex_error</code>	1304

31.7	Class template <code>regex_traits</code> . . . . .	1305
31.8	Class template <code>basic_regex</code> . . . . .	1307
31.9	Class template <code>sub_match</code> . . . . .	1313
31.10	Class template <code>match_results</code> . . . . .	1318
31.11	Regular expression algorithms . . . . .	1324
31.12	Regular expression iterators . . . . .	1329
31.13	Modified ECMAScript regular expression grammar . . . . .	1335
<b>32</b>	<b>Atomic operations library</b>	<b>1338</b>
32.1	General . . . . .	1338
32.2	Header <code>&lt;atomic&gt;</code> synopsis . . . . .	1338
32.3	Type aliases . . . . .	1342
32.4	Order and consistency . . . . .	1342
32.5	Lock-free property . . . . .	1344
32.6	Class template <code>atomic</code> . . . . .	1345
32.7	Non-member functions . . . . .	1352
32.8	Flag type and operations . . . . .	1352
32.9	Fences . . . . .	1353
<b>33</b>	<b>Thread support library</b>	<b>1355</b>
33.1	General . . . . .	1355
33.2	Requirements . . . . .	1355
33.3	Threads . . . . .	1358
33.4	Mutual exclusion . . . . .	1363
33.5	Condition variables . . . . .	1384
33.6	Futures . . . . .	1391
<b>A</b>	<b>Grammar summary</b>	<b>1408</b>
A.1	Keywords . . . . .	1408
A.2	Lexical conventions . . . . .	1408
A.3	Basic concepts . . . . .	1413
A.4	Expressions . . . . .	1413
A.5	Statements . . . . .	1417
A.6	Declarations . . . . .	1418
A.7	Declarators . . . . .	1422
A.8	Classes . . . . .	1424
A.9	Derived classes . . . . .	1425
A.10	Special member functions . . . . .	1426
A.11	Overloading . . . . .	1426
A.12	Templates . . . . .	1426
A.13	Exception handling . . . . .	1427
A.14	Preprocessing directives . . . . .	1428
<b>B</b>	<b>Implementation quantities</b>	<b>1430</b>
<b>C</b>	<b>Compatibility</b>	<b>1432</b>
C.1	C++ and ISO C . . . . .	1432
C.2	C++ and ISO C++ 2003 . . . . .	1441
C.3	C++ and ISO C++ 2011 . . . . .	1447
C.4	C++ and ISO C++ 2014 . . . . .	1449
C.5	C standard library . . . . .	1453

<b>D Compatibility features</b>	<b>1456</b>
D.1 Redeclaration of <code>static constexpr</code> data members . . . . .	1456
D.2 Implicit declaration of copy functions . . . . .	1456
D.3 Deprecated exception specifications . . . . .	1456
D.4 C++ standard library headers . . . . .	1456
D.5 C standard library headers . . . . .	1457
D.6 <code>char*</code> streams . . . . .	1457
D.7 <code>uncaught_exception</code> . . . . .	1466
D.8 Old adaptable function bindings . . . . .	1466
D.9 The default allocator . . . . .	1471
D.10 Raw storage iterator . . . . .	1472
D.11 Temporary buffers . . . . .	1473
D.12 Deprecated type traits . . . . .	1474
D.13 Deprecated iterator primitives . . . . .	1475
D.14 Deprecated <code>shared_ptr</code> observers . . . . .	1475
D.15 Deprecated standard code conversion facets . . . . .	1475
D.16 Deprecated convenience conversion interfaces . . . . .	1477
<b>Bibliography</b>	<b>1482</b>
<b>Cross references</b>	<b>1483</b>
<b>Cross references from ISO C++ 2014</b>	<b>1504</b>
<b>Index</b>	<b>1506</b>
<b>Index of grammar productions</b>	<b>1539</b>
<b>Index of library names</b>	<b>1543</b>
<b>Index of implementation-defined behavior</b>	<b>1601</b>

## Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 ISO documents 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](http://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 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](http://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 voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html).

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

This fifth edition cancels and replaces the fourth edition (ISO/IEC 14882:2014), which has been technically revised.

The main changes compared to the previous edition are as follows:

- expression evaluation order is specified in more cases
- removal of trigraphs
- adjustments to value categories resulting in copy elision being mandatory
- additional character and floating point literal syntaxes
- lambda expressions extended to permit capture of `*this` and use in constant expressions
- initializer statements for `if` and `switch` statements
- addition of `constexpr` if statements
- range-based `for` statement generalized to support heterogeneous `begin` and `end` types
- addition of structured bindings
- addition of inline variables
- list initialization extended to support enumerations and aggregates with base classes
- message in `static_assert` is now optional
- addition of nested namespace definition syntax

- extended support for attributes
- exception specifications are now part of function types
- template argument deduction is now supported for class templates
- addition of fold expressions
- pack expansion can be performed on using declarations
- permitted forms of template parameters and template arguments have been generalized
- dynamic allocation is supported for over-aligned types
- preprocessor can detect presence of header files with `__has_include`
- new utility functions, types, and templates in the standard library, including
  - an `any` type
  - an `optional` class template
  - a `variant` class template
  - a `clamp` function
  - a `std::byte` type
  - a `not_fn` function
  - a `void_t` alias template
  - `conjunction`, `disjunction`, and `negation` templates
  - an `invoke` function, and `is_invocable` and `invoke_result` type traits
  - an `is_swappable` type trait
- extended constant expression evaluation support in the standard library
- elementary conversion functions between strings and numeric types added
- constructors for `pair` and `tuple` are conditionally-explicit
- `shared_ptrs` of array types now supported
- additional algorithms for managing uninitialized memory
- addition of polymorphic memory resources
- addition of substring search facilities providing the Boyer-Moore and Boyer-Moore-Horspool search algorithms
- addition of variable templates for type traits
- addition of a non-owning string view template
- ability to splice elements between containers for maps and sets
- better support for element insertion in unique-key maps
- support for incomplete types in containers
- addition of parallel algorithms
- addition of `sample` algorithm
- addition of mathematical special functions, and `gcd`, `lcm`, and three-argument `hypot` functions
- addition of support for operations on file systems
- addition of shared mutexes and variadic lock guards
- removal of deprecated features