

DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 14882

ISO/IEC JTC 1/SC 22

Secretariat: ANSI

Voting begins on:
2020-06-11

Voting terminates on:
2020-09-03

Programming languages — C++

Langages de programmation — C++

ICS: 35.060

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/3ee2b5bc-5de4-4642-8669-a9c9ef29d05e/iso-iec-dis-14882>

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document is circulated as received from the committee secretariat.



Reference number
ISO/IEC DIS 14882:2020(E)

© ISO/IEC 2020

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
<https://standards.iteh.ai/catalog/standards/sist/3ee2b5bc-5de4-4642-8669-a9c9ef29d05e/iso-iec-dis-14882>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, 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
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword	x
1 Scope	1
2 Normative references	2
3 Terms and definitions	3
4 General principles	7
4.1 Implementation compliance	7
4.2 Structure of this document	8
4.3 Syntax notation	8
4.4 Acknowledgments	9
5 Lexical conventions	10
5.1 Separate translation	10
5.2 Phases of translation	10
5.3 Character sets	11
5.4 Preprocessing tokens	12
5.5 Alternative tokens	13
5.6 Tokens	13
5.7 Comments	13
5.8 Header names	14
5.9 Preprocessing numbers	14
5.10 Identifiers	14
5.11 Keywords	15
5.12 Operators and punctuators	16
5.13 Literals	16
6 Basics	26
6.1 Preamble	26
6.2 Declarations and definitions	26
6.3 One-definition rule	28
6.4 Scope	32
6.5 Name lookup	38
6.6 Program and linkage	51
6.7 Memory and objects	54
6.8 Types	68
6.9 Program execution	74
7 Expressions	86
7.1 Preamble	86
7.2 Properties of expressions	87
7.3 Standard conversions	89
7.4 Usual arithmetic conversions	94
7.5 Primary expressions	95
7.6 Compound expressions	110
7.7 Constant expressions	140
8 Statements	146
8.1 Preamble	146
8.2 Labeled statement	147
8.3 Expression statement	147
8.4 Compound statement or block	147
8.5 Selection statements	147

8.6	Iteration statements	149
8.7	Jump statements	152
8.8	Declaration statement	153
8.9	Ambiguity resolution	154
9	Declarations	156
9.1	Preamble	156
9.2	Specifiers	157
9.3	Declarators	174
9.4	Initializers	188
9.5	Function definitions	204
9.6	Structured binding declarations	210
9.7	Enumerations	211
9.8	Namespaces	214
9.9	The <code>using</code> declaration	220
9.10	The <code>asm</code> declaration	226
9.11	Linkage specifications	226
9.12	Attributes	228
10	Modules	236
10.1	Module units and purviews	236
10.2	Export declaration	237
10.3	Import declaration	240
10.4	Global module fragment	241
10.5	Private module fragment	243
10.6	Instantiation context	244
10.7	Reachability	245
11	Classes	247
11.1	Preamble	247
11.2	Properties of classes	248
11.3	Class names	249
11.4	Class members	250
11.5	Unions	272
11.6	Local class declarations	274
11.7	Derived classes	275
11.8	Member name lookup	283
11.9	Member access control	285
11.10	Initialization	294
11.11	Comparisons	306
11.12	Free store	308
12	Overloading	311
12.1	Preamble	311
12.2	Overloadable declarations	311
12.3	Declaration matching	313
12.4	Overload resolution	314
12.5	Address of overloaded function	337
12.6	Overloaded operators	338
12.7	Built-in operators	341
12.8	User-defined literals	344
13	Templates	346
13.1	Preamble	346
13.2	Template parameters	347
13.3	Names of template specializations	351
13.4	Template arguments	353
13.5	Template constraints	359
13.6	Type equivalence	364

13.7	Template declarations	364
13.8	Name resolution	385
13.9	Template instantiation and specialization	401
13.10	Function template specializations	413
14	Exception handling	433
14.1	Preamble	433
14.2	Throwing an exception	434
14.3	Constructors and destructors	435
14.4	Handling an exception	436
14.5	Exception specifications	437
14.6	Special functions	440
15	Preprocessing directives	442
15.1	Preamble	442
15.2	Conditional inclusion	444
15.3	Source file inclusion	446
15.4	Module directive	447
15.5	Header unit importation	447
15.6	Macro replacement	449
15.7	Line control	454
15.8	Error directive	454
15.9	Pragma directive	454
15.10	Null directive	455
15.11	Predefined macro names	455
15.12	Pragma operator	457
16	Library introduction	458
16.1	General	458
16.2	The C standard library	459
16.3	Definitions	459
16.4	Method of description	462
16.5	Library-wide requirements	468
17	Language support library	488
17.1	General	488
17.2	Common definitions	488
17.3	Implementation properties	492
17.4	Integer types	502
17.5	Startup and termination	503
17.6	Dynamic memory management	504
17.7	Type identification	511
17.8	Source location	513
17.9	Exception handling	515
17.10	Initializer lists	519
17.11	Comparisons	520
17.12	Coroutines	527
17.13	Other runtime support	532
18	Concepts library	534
18.1	General	534
18.2	Equality preservation	534
18.3	Header <concepts> synopsis	535
18.4	Language-related concepts	537
18.5	Comparison concepts	542
18.6	Object concepts	545
18.7	Callable concepts	545

19	Diagnostics library	547
19.1	General	547
19.2	Exception classes	547
19.3	Assertions	550
19.4	Error numbers	550
19.5	System error support	552
20	General utilities library	561
20.1	General	561
20.2	Utility components	561
20.3	Compile-time integer sequences	566
20.4	Pairs	567
20.5	Tuples	571
20.6	Optional objects	581
20.7	Variants	593
20.8	Storage for any type	604
20.9	Bitsets	609
20.10	Memory	614
20.11	Smart pointers	630
20.12	Memory resources	653
20.13	Class template <code>scoped_allocator_adaptor</code>	662
20.14	Function objects	666
20.15	Metaprogramming and type traits	689
20.16	Compile-time rational arithmetic	713
20.17	Class <code>type_index</code>	715
20.18	Execution policies	717
20.19	Primitive numeric conversions	718
20.20	Formatting	721
21	Strings library	739
21.1	General	739
21.2	Character traits	739
21.3	String classes	744
21.4	String view classes	770
21.5	Null-terminated sequence utilities	779
22	Containers library	785
22.1	General	785
22.2	Container requirements	785
22.3	Sequence containers	818
22.4	Associative containers	846
22.5	Unordered associative containers	864
22.6	Container adaptors	886
22.7	Views	894
23	Iterators library	901
23.1	General	901
23.2	Header <code><iterator></code> synopsis	901
23.3	Iterator requirements	908
23.4	Iterator primitives	928
23.5	Iterator adaptors	931
23.6	Stream iterators	952
23.7	Range access	957
24	Ranges library	960
24.1	General	960
24.2	Header <code><ranges></code> synopsis	960
24.3	Range access	965
24.4	Range requirements	969

24.5	Range utilities	972
24.6	Range factories	977
24.7	Range adaptors	987
25	Algorithms library	1024
25.1	General	1024
25.2	Algorithms requirements	1024
25.3	Parallel algorithms	1026
25.4	Header <code><algorithm></code> synopsis	1029
25.5	Algorithm result types	1064
25.6	Non-modifying sequence operations	1066
25.7	Mutating sequence operations	1078
25.8	Sorting and related operations	1094
25.9	Header <code><numeric></code> synopsis	1121
25.10	Generalized numeric operations	1124
25.11	Specialized <code><memory></code> algorithms	1134
25.12	C library algorithms	1140
26	Numerics library	1141
26.1	General	1141
26.2	Numeric type requirements	1141
26.3	The floating-point environment	1141
26.4	Complex numbers	1142
26.5	Bit manipulation	1150
26.6	Random number generation	1153
26.7	Numeric arrays	1190
26.8	Mathematical functions for floating-point types	1209
26.9	Numbers	1224
27	Time library	1225
27.1	General	1225
27.2	Header <code><chrono></code> synopsis	1225
27.3	<i>Cpp17Clock</i> requirements	1239
27.4	Time-related traits	1239
27.5	Class template <code>duration</code>	1241
27.6	Class template <code>time_point</code>	1248
27.7	Clocks	1250
27.8	The civil calendar	1261
27.9	Class template <code>hh_mm_ss</code>	1290
27.10	12/24 hours functions	1293
27.11	Time zones	1293
27.12	Formatting	1306
27.13	Parsing	1310
27.14	Header <code><ctime></code> synopsis	1313
28	Localization library	1315
28.1	General	1315
28.2	Header <code><locale></code> synopsis	1315
28.3	Locales	1316
28.4	Standard <code>locale</code> categories	1322
28.5	C library locales	1353
29	Input/output library	1355
29.1	General	1355
29.2	Iostreams requirements	1355
29.3	Forward declarations	1356
29.4	Standard istream objects	1358
29.5	Iostreams base classes	1359
29.6	Stream buffers	1375

29.7	Formatting and manipulators	1383
29.8	String-based streams	1407
29.9	File-based streams	1421
29.10	Synchronized output streams	1433
29.11	File systems	1437
29.12	C library files	1482
30	Regular expressions library	1486
30.1	General	1486
30.2	Definitions	1486
30.3	Requirements	1487
30.4	Header <code><regex></code> synopsis	1488
30.5	Namespace <code>std::regex_constants</code>	1492
30.6	Class <code>regex_error</code>	1495
30.7	Class template <code>regex_traits</code>	1495
30.8	Class template <code>basic_regex</code>	1497
30.9	Class template <code>sub_match</code>	1501
30.10	Class template <code>match_results</code>	1503
30.11	Regular expression algorithms	1508
30.12	Regular expression iterators	1512
30.13	Modified ECMAScript regular expression grammar	1517
31	Atomic operations library	1520
31.1	General	1520
31.2	Header <code><atomic></code> synopsis	1520
31.3	Type aliases	1524
31.4	Order and consistency	1524
31.5	Lock-free property	1526
31.6	Waiting and notifying	1526
31.7	Class template <code>atomic_ref</code>	1527
31.8	Class template <code>atomic</code>	1533
31.9	Non-member functions	1548
31.10	Flag type and operations	1548
31.11	Fences	1550
32	Thread support library	1551
32.1	General	1551
32.2	Requirements	1551
32.3	Stop tokens	1553
32.4	Threads	1558
32.5	Mutual exclusion	1565
32.6	Condition variables	1583
32.7	Semaphore	1590
32.8	Coordination types	1592
32.9	Futures	1595
A	Grammar summary	1610
A.1	Keywords	1610
A.2	Lexical conventions	1610
A.3	Basics	1614
A.4	Expressions	1614
A.5	Statements	1618
A.6	Declarations	1619
A.7	Modules	1625
A.8	Classes	1626
A.9	Overloading	1627
A.10	Templates	1627
A.11	Exception handling	1629
A.12	Preprocessing directives	1629

B Implementation quantities	1631
C Compatibility	1633
C.1 C++ and ISO C++ 2017	1633
C.2 C++ and ISO C++ 2014	1640
C.3 C++ and ISO C++ 2011	1643
C.4 C++ and ISO C++ 2003	1645
C.5 C++ and ISO C	1650
C.6 C standard library	1658
D Compatibility features	1661
D.1 Arithmetic conversion on enumerations	1661
D.2 Implicit capture of <code>*this</code> by reference	1661
D.3 Comma operator in subscript expressions	1661
D.4 Array comparisons	1661
D.5 Deprecated <code>volatile</code> types	1662
D.6 Redclaration of <code>static constexpr</code> data members	1662
D.7 Non-local use of TU-local entities	1662
D.8 Implicit declaration of copy functions	1663
D.9 C headers	1663
D.10 Requires paragraph	1664
D.11 Relational operators	1664
D.12 <code>char*</code> streams	1664
D.13 Deprecated type traits	1672
D.14 Tuple	1672
D.15 Variant	1673
D.16 Deprecated iterator primitives	1673
D.17 Deprecated <code>move_iterator</code> access	1674
D.18 Deprecated <code>shared_ptr</code> atomic access	1674
D.19 Deprecated <code>basic_string</code> capacity	1676
D.20 Deprecated standard code conversion facets	1676
D.21 Deprecated convenience conversion interfaces	1677
D.22 Deprecated locale category facets	1681
D.23 Deprecated filesystem path factory functions	1681
D.24 Deprecated atomic operations	1682
Bibliography	1684
Cross references	1685
Cross references from ISO C++ 2017	1707
Index	1710
Index of grammar productions	1744
Index of library headers	1749
Index of library names	1751
Index of library concepts	1823
Index of implementation-defined behavior	1826

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.

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 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).

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) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

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.

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

This sixth edition cancels and replaces the fifth edition (ISO/IEC 14882:2017), which has been technically revised. It also incorporates the Technical Specifications

- ISO/IEC TS 19217:2015 *Programming Languages — C++ Extensions for concepts*,
- ISO/IEC TS 21425:2017 *Programming Languages — C++ Extensions for ranges*,
- ISO/IEC TS 22277:2017 *Programming Languages — C++ Extensions for Coroutines*,
- ISO/IEC TS 21544:2018 *Programming Languages — Extensions to C++ for Modules*,
- portions of ISO/IEC TS 19571:2016 *Programming Languages — Technical specification for C++ extensions for concurrency*, and
- portions of ISO/IEC TS 19568:2017 *Programming Languages — C++ Extensions for Library Fundamentals*.

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

- addition of concepts, *requires-clauses*, and *requires-expressions* and `<concepts>` (18.3) header
- addition of coroutines, including `co_yield`, `co_await`, and `co_return` keywords and `<coroutine>` (17.12.1) header
- addition of modules, *import-declarations*, and *export-declarations*
- addition of three-way comparison, defaulted comparisons, rewriting of comparison operator expressions, and `<compare>` (17.11.1) header
- addition of designated initializers
- support for class types and floating-point types as the type of a non-type template parameter
- new attributes `[[no_unique_address]]`, `[[likely]]`, `[[unlikely]]`
- support for optional reason string in `[[nodiscard]]` attribute
- ability to require constant initialization with `constexpr` keyword

- ability to require constant evaluation with `constexpr` keyword
- extension of constant evaluation support to cover
 - memory allocation and deallocation
 - virtual function calls
 - `dynamic_cast` and `typeid`
 - changing the active member of a union
 - uninitialized variables
- support for `constexpr` functions to contain `catch` handlers and *asm-declarations* that are not reached during constant evaluation
- support for controlling destruction in a class-specific operator delete function
- addition of `using enum` declaration
- addition of `char8_t` type
- guarantee that `char16_t` and `char32_t` literals are encoded as UTF-16 and UTF-32 respectively
- support for an initializer statement in range-based for loops
- support for default member initializers for bit-fields
- support for parenthesized aggregate initialization
- extended support for lambda expressions, including
 - explicit *template-heads* in generic lambdas
 - default construction and assignment of stateless closure types
 - lambda expressions in unevaluated operands
 - pack expansion of lambda *init-captures*
- generalized support for structured bindings
- support for inline namespaces in nested namespace definitions
- support for conditionally-explicit member functions
- extended support for class template argument deduction to cover aggregate initialization and alias templates
- reduced cases in which `typename` is required
- support for calling an undeclared *template-id* via argument-dependent name lookup
- relaxed access checking rules in template specialization declarations
- revised memory model
- expanded cases in which returned or thrown variables are implicitly moved
- extended support for variadic macros with `__VA_OPT__`
- feature test macros and `<version>` (17.3.2) header
- restricted valid uses of standard library functions and function templates
- addition of ranges and `<ranges>` (24.2) header
- addition of calendar and time zone support
- addition of text formatting library and `<format>` (20.20.1) header
- addition of synchronization facilities for waiting, notifications, semaphores, latches, and barriers, and `<barrier>` (32.8.2.1), `<latch>` (32.8.1.1) and `<semaphore>` (32.7.1) headers
- addition of mathematical constants library and `<numbers>` (26.9.1) header
- support for representing source locations and `<source_location>` (17.8.1) header
- addition of `span` view and `<span>` (22.7.2) header
- addition of joining thread class and `<stop_token>` (32.3.2) header
- extensions to atomic types and operations, including

- new class template `atomic_ref`
- new atomic floating-point operations
- new atomic smart pointer types
- support for compare-and-exchange operations on types with padding
- conversion of `memory_order` to a scoped enumeration
- addition of `unsequenced` execution policy
- new utility functions, types, and templates in the standard library, including
 - new type traits `is_bounded_array`, `is_corresponding_member`, `is_layout_compatible`, `is_nothrow_convertible`, `is_pointer_interconvertible_base_of`, `is_pointer_interconvertible_with_class`, and `is_unbounded_array`
 - new transformation traits `remove_cvref`, `type_identity`, `unwrap_ref_decay`, and `unwrap_reference`
 - new standard library primitive functions `assume_aligned` and `is_constant_evaluated`
 - new free functions `erase` and `erase_if` for containers
 - utilities for uses-allocator construction
 - function template `to_address`
 - function template `bind_front`
 - function template `to_array`
- `make_shared` support for array types
- support for allocating objects owned by `shared_ptr` and `weak_ptr` with default initialization
- addition of bit manipulation library and `<bit>` (26.5.2) header
- addition of a synchronized buffered output stream and `<syncstream>` (29.10.1) header
- addition of `shift` algorithms
- addition of `midpoint` and `lerp` math functions
- use of `[[nodiscard]]` attribute in the standard library
- support for heterogeneous lookup for unordered containers
- support for element existence detection in associative containers
- change to return removed element count from `remove`, `remove_if`, and `unique` member functions on `list` and `forward_list`
- addition of `starts_with` and `ends_with` to `basic_string` and `basic_string_view`
- support for move semantics in `<numeric>` (25.9) algorithms
- support for efficient access to the buffer of a `basic_stringbuf`
- extended constant expression evaluation support in the standard library to cover more algorithms, utilities, and types, including `pair`, `tuple`, `vector`, and `string`
- removal of deprecated features

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

1 Scope

[intro.scope]

- ¹ This document specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements and relaxations of the first requirement appear at various places within this document.
- ² C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2018 *Programming languages — C* (hereinafter referred to as the *C standard*). C++ provides many facilities beyond those provided by C, including additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

iTeh STANDARD PREVIEW
(standards.itih.ai)
Full standard:
<https://standards.itih.ai/catalog/standards/sist/3ee2b5bc-5de4-4642-8669-a9c9ef29d05e/iso-iec-dis-14882>