

---

---

**Programming languages — C++**

*Langages de programmation — C++*

**iTeh STANDARD PREVIEW**  
**(standards.iteh.ai)**

[ISO/IEC 14882:2003](https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-aeb0-59dc13467ca9/iso-iec-14882-2003)

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-aeb0-59dc13467ca9/iso-iec-14882-2003>

**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.

**iTeh STANDARD PREVIEW**  
**(standards.iteh.ai)**

[ISO/IEC 14882:2003](#)

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-aeb0-59dc13467ca9/iso-iec-14882-2003>

© ISO/IEC 2003

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](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Published in Switzerland

# Contents

1	General.....	1
1.1	Scope.....	1
1.2	Normative references .....	1
1.3	Terms and definitions .....	1
1.3.1	argument .....	1
1.3.2	diagnostic message .....	2
1.3.3	dynamic type.....	2
1.3.4	ill-formed program.....	2
1.3.5	implementation-defined behavior .....	2
1.3.6	implementation limits.....	2
1.3.7	locale-specific behavior .....	2
1.3.8	multibyte character .....	2
1.3.9	parameter.....	2
1.3.10	signature.....	2
1.3.11	static type.....	2
1.3.12	undefined behavior .....	2
1.3.13	unspecified behavior.....	3
1.3.14	well-formed program .....	3
1.4	Implementation compliance.....	3
1.5	Structure of this International Standard.....	4
1.6	Syntax notation .....	4
1.7	The C++ memory model.....	4
1.8	The C++ object model .....	4
1.9	Program execution .....	5

1.10	Acknowledgments .....	8
2	Lexical conventions .....	9
2.1	Phases of translation .....	9
2.2	Character sets .....	10
2.3	Trigraph sequences .....	11
2.4	Preprocessing tokens .....	11
2.5	Alternative tokens .....	12
2.6	Tokens .....	12
2.7	Comments .....	12
2.8	Header names .....	13
2.9	Preprocessing numbers .....	13
2.10	Identifiers .....	13
2.11	Keywords .....	14
2.12	Operators and punctuators .....	15
2.13	Literals .....	15
2.13.1	Integer literals .....	15
2.13.2	Character literals .....	16
2.13.3	Floating literals .....	18
2.13.4	String literals .....	19
2.13.5	Boolean literals .....	19
3	Basic concepts .....	21
3.1	Declarations and definitions .....	21
3.2	One definition rule .....	22
3.3	Declarative regions and scopes .....	24
3.3.1	Point of declaration .....	25
3.3.2	Local scope .....	26
3.3.3	Function prototype scope .....	26
3.3.4	Function scope .....	27
3.3.5	Namespace scope .....	27
3.3.6	Class scope .....	27
3.3.7	Name hiding .....	28
3.4	Name lookup .....	29
3.4.1	Unqualified name lookup .....	29
3.4.2	Argument-dependent name lookup .....	32
3.4.3	Qualified name lookup .....	34

**ITeH STANDARD PREVIEW**  
**(standards.iteh.ai)**

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae05-59dc13467ca9/iso-iec-14882-2003>

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae05-59dc13467ca9/iso-iec-14882-2003>

3.4.3.1	Class members .....	35
3.4.3.2	Namespace members .....	35
3.4.4	Elaborated type specifiers .....	39
3.4.5	Class member access .....	40
3.4.6	Using-directives and namespace aliases .....	41
3.5	Program and linkage .....	41
3.6	Start and termination .....	43
3.6.1	Main function .....	43
3.6.2	Initialization of non-local objects .....	44
3.6.3	Termination .....	45
3.7	Storage duration .....	46
3.7.1	Static storage duration .....	46
3.7.2	Automatic storage duration .....	46
3.7.3	Dynamic storage duration .....	47
3.7.3.1	Allocation functions .....	47
3.7.3.2	Deallocation functions .....	48
3.7.4	Duration of sub-objects .....	48
3.8	Object Lifetime .....	49
3.9	Types .....	52
3.9.1	Fundamental types .....	53
3.9.2	Compound types .....	55
3.9.3	CV-qualifiers .....	55
3.10	Lvalues and rvalues .....	56
4	Standard conversions .....	59
4.1	Lvalue-to-rvalue conversion .....	59
4.2	Array-to-pointer conversion .....	60
4.3	Function-to-pointer conversion .....	60
4.4	Qualification conversions .....	60
4.5	Integral promotions .....	61
4.6	Floating point promotion .....	61
4.7	Integral conversions .....	62
4.8	Floating point conversions .....	62
4.9	Floating-integral conversions .....	62
4.10	Pointer conversions .....	62
4.11	Pointer to member conversions .....	63

**STANDARD PREVIEW**

**(standards.iteh.ai)**

[ISO/IEC 14882:2003](https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae0-59dc13467ca9/iso-iec-14882-2003)

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae0-59dc13467ca9/iso-iec-14882-2003>

4.12 Boolean conversions .....63

5 Expressions .....65

5.1 Primary expressions .....66

5.2 Postfix expressions .....68

5.2.1 Subscripting .....68

5.2.2 Function call .....68

5.2.3 Explicit type conversion (functional notation) .....70

5.2.4 Pseudo destructor call .....70

5.2.5 Class member access .....70

5.2.6 Increment and decrement .....71

5.2.7 Dynamic cast .....72

5.2.8 Type identification .....73

5.2.9 Static cast .....74

5.2.10 Reinterpret cast .....75

5.2.11 Const cast .....76

5.3 Unary expressions .....78

5.3.1 Unary operators .....78

5.3.2 Increment and decrement .....79

5.3.3 sizeof .....79

5.3.4 New .....80

5.3.5 Delete .....83

5.4 Explicit type conversion (cast notation) .....84

5.5 Pointer-to-member operators .....85

5.6 Multiplicative operators .....85

5.7 Additive operators .....86

5.8 Shift operators .....87

5.9 Relational operators .....87

5.10 Equality operators .....88

5.11 Bitwise AND operator .....89

5.12 Bitwise exclusive OR operator .....89

5.13 Bitwise inclusive OR operator .....89

5.14 Logical AND operator .....89

5.15 Logical OR operator .....90

5.16 Conditional operator .....90

5.17 Assignment operators .....91

ITeH STANDARD PREVIEW  
(standards.iteh.ai)

ISO/IEC 14882:2003  
iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae0-59dc13467ca9/iso-iec-14882-2003

5.18	Comma operator .....	92
5.19	Constant expressions .....	92
6	Statements.....	95
6.1	Labeled statement .....	95
6.2	Expression statement .....	95
6.3	Compound statement or block .....	95
6.4	Selection statements.....	96
6.4.1	The if statement.....	97
6.4.2	The switch statement .....	97
6.5	Iteration statements .....	97
6.5.1	The while statement.....	98
6.5.2	The do statement .....	98
6.5.3	The for statement.....	99
6.6	Jump statements.....	99
6.6.1	The break statement.....	99
6.6.2	The continue statement.....	100
6.6.3	The return statement .....	100
6.6.4	The goto statement .....	100
6.7	Declaration statement.....	100
6.8	Ambiguity resolution .....	101
7	Declarations .....	103
7.1	Specifiers .....	104
7.1.1	Storage class specifiers .....	105
7.1.2	Function specifiers.....	106
7.1.3	The typedef specifier.....	107
7.1.4	The friend specifier.....	108
7.1.5	Type specifiers.....	108
7.1.5.1	The <i>cv-qualifiers</i> .....	109
7.1.5.2	Simple type specifiers.....	110
7.1.5.3	Elaborated type specifiers .....	111
7.2	Enumeration declarations .....	112
7.3	Namespaces .....	114
7.3.1	Namespace definition .....	114
7.3.1.1	Unnamed namespaces.....	115
7.3.1.2	Namespace member definitions.....	115
7.3.2	Namespace alias.....	117
7.3.3	The using declaration .....	117
7.3.4	Using directive.....	123
7.4	The asm declaration .....	126

7.5 Linkage specifications .....126

8 Declarators .....131

8.1 Type names .....132

8.2 Ambiguity resolution .....132

8.3 Meaning of declarators .....134

8.3.1 Pointers .....135

8.3.2 References .....135

8.3.3 Pointers to members .....136

8.3.4 Arrays .....137

8.3.5 Functions .....138

8.3.6 Default arguments .....141

8.4 Function definitions .....144

8.5 Initializers .....145

8.5.1 Aggregates .....147

8.5.2 Character arrays .....150

8.5.3 References .....150

9 Classes .....153

9.1 Class names .....153

9.2 Class members .....155

9.3 Member functions .....157

9.3.1 Nonstatic member functions .....158

9.3.2 The `this` pointer .....160

9.4 Static members .....160

9.4.1 Static member functions .....161

9.4.2 Static data members .....161

9.5 Unions .....162

9.6 Bit-fields .....163

9.7 Nested class declarations .....164

9.8 Local class declarations .....165

9.9 Nested type names .....166

10 Derived classes .....167

10.1 Multiple base classes .....168

10.2 Member name lookup .....169

10.3 Virtual functions .....172

ITeH STANDARD PREVIEW  
(standards.iteh.ai)

ISO/IEC 14882:2003  
<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-aeb0-59dc13467ca9/iso-iec-14882-2003>



10.4	Abstract classes.....	176
11	Member access control .....	179
11.1	Access specifiers.....	180
11.2	Accessibility of base classes and base class members.....	181
11.3	Access declarations.....	182
11.4	Friends .....	183
11.5	Protected member access .....	186
11.6	Access to virtual functions.....	187
11.7	Multiple access .....	188
11.8	Nested classes .....	188
12	Special member functions.....	189
12.1	Constructors.....	189
12.2	Temporary objects .....	191
12.3	Conversions.....	192
12.3.1	Conversion by constructor.....	193
12.3.2	Conversion functions.....	194
12.4	Destructors.....	195
12.5	Free store .....	198
12.6	Initialization.....	199
12.6.1	Explicit initialization .....	200
12.6.2	Initializing bases and members.....	201
12.7	Construction and destruction .....	204
12.8	Copying class objects .....	207
13	Overloading .....	213
13.1	Overloadable declarations.....	213
13.2	Declaration matching.....	215
13.3	Overload resolution .....	216
13.3.1	Candidate functions and argument lists.....	217
13.3.1.1	Function call syntax.....	218
13.3.1.1.1	Call to named function.....	218
13.3.1.1.2	Call to object of class type.....	219
13.3.1.2	Operators in expressions.....	220

13.3.1.3 Initialization by constructor .....222

13.3.1.4 Copy-initialization of class by user-defined conversion.....222

13.3.1.5 Initialization by conversion function .....222

13.3.1.6 Initialization by conversion function for direct reference binding .....223

13.3.2 Viable functions.....223

13.3.3 Best Viable Function .....223

13.3.3.1 Implicit conversion sequences .....225

13.3.3.1.1 Standard conversion sequences .....227

13.3.3.1.2 User-defined conversion sequences.....227

13.3.3.1.3 Ellipsis conversion sequences.....228

13.3.3.1.4 Reference binding.....228

13.3.3.2 Ranking implicit conversion sequences.....228

13.4 Address of overloaded function .....230

13.5 Overloaded operators.....232

13.5.1 Unary operators.....233

13.5.2 Binary operators.....233

13.5.3 Assignment .....233

13.5.4 Function call .....234

13.5.5 Subscripting .....234

13.5.6 Class member access .....234

13.5.7 Increment and decrement.....234

13.6 Built-in operators .....235

14 Templates.....239

14.1 Template parameters.....240

14.2 Names of template specializations.....242

14.3 Template arguments.....244

14.3.1 Template type arguments.....245

14.3.2 Template non-type arguments .....246

14.3.3 Template template arguments.....248

14.4 Type equivalence .....248

14.5 Template declarations .....249

14.5.1 Class templates .....249

14.5.1.1 Member functions of class templates.....249

14.5.1.2 Member classes of class templates .....250

14.5.1.3 Static data members of class templates.....250

14.5.2 Member templates .....251

14.5.3 Friends .....252

14.5.4 Class template partial specializations .....254

14.5.4.1 Matching of class template partial specializations .....256

14.5.4.2 Partial ordering of class template specializations .....257

14.5.4.3 Members of class template specializations.....257

14.5.5 Function templates.....258

14.5.5.1 Function template overloading .....259

14.5.5.2 Partial ordering of function templates .....260

ITeH STANDARD PREVIEW  
(standards.iteh.ai)

ISO/IEC 14882:2003

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae05-59dc13467ca9/iso-iec-14882-2003>

14.6	Name resolution .....	261
14.6.1	Locally declared names .....	264
14.6.2	Dependent names .....	267
14.6.2.1	Dependent types .....	268
14.6.2.2	Type-dependent expressions .....	268
14.6.2.3	Value-dependent expressions .....	269
14.6.2.4	Dependent template arguments .....	269
14.6.3	Non-dependent names .....	270
14.6.4	Dependent name resolution .....	270
14.6.4.1	Point of instantiation .....	270
14.6.4.2	Candidate functions .....	271
14.6.5	Friend names declared within a class template .....	271
14.7	Template instantiation and specialization .....	272
14.7.1	Implicit instantiation .....	273
14.7.2	Explicit instantiation .....	276
14.7.3	Explicit specialization .....	277
14.8	Function template specializations .....	282
14.8.1	Explicit template argument specification .....	283
14.8.2	Template argument deduction .....	285
14.8.2.1	Deducing template arguments from a function call .....	287
14.8.2.2	Deducing template arguments taking the address of a function template .....	288
14.8.2.3	Deducing conversion function template arguments .....	288
14.8.2.4	Deducing template arguments from a type .....	288
14.8.3	Overload resolution .....	293
15	Exception handling .....	297
15.1	Throwing an exception .....	298
15.2	Constructors and destructors .....	300
15.3	Handling an exception .....	300
15.4	Exception specifications .....	302
15.5	Special functions .....	304
15.5.1	The <code>terminate()</code> function .....	304
15.5.2	The <code>unexpected()</code> function .....	305
15.5.3	The <code>uncaught_exception()</code> function .....	305
15.6	Exceptions and access .....	305
16	Preprocessing directives .....	307
16.1	Conditional inclusion .....	308
16.2	Source file inclusion .....	309
16.3	Macro replacement .....	310
16.3.1	Argument substitution .....	311
16.3.2	The <code>#</code> operator .....	311
16.3.3	The <code>##</code> operator .....	312

16.3.4	Rescanning and further replacement.....	312
16.3.5	Scope of macro definitions .....	312
16.4	Line control.....	314
16.5	Error directive .....	314
16.6	Pragma directive .....	314
16.7	Null directive .....	314
16.8	Predefined macro names.....	315
17	Library introduction.....	317
17.1	Definitions .....	317
17.1.1	arbitrary-positional stream.....	317
17.1.2	character.....	317
17.1.3	character container type.....	317
17.1.4	comparison function .....	317
17.1.5	component.....	318
17.1.6	default behavior .....	318
17.1.7	handler function.....	318
17.1.8	iostream class templates .....	318
17.1.9	modifier function .....	318
17.1.10	object state .....	318
17.1.11	narrow-oriented iostream classes.....	318
17.1.12	NTCTS.....	318
17.1.13	observer function .....	318
17.1.14	replacement function.....	318
17.1.15	required behavior .....	318
17.1.16	repositional stream.....	319
17.1.17	reserved function.....	319
17.1.18	traits class.....	319
17.1.19	wide-oriented iostream classes .....	319
17.2	Additional definitions .....	319
17.3	Method of description (Informative) .....	319
17.3.1	Structure of each subclause.....	319
17.3.1.1	Summary.....	320
17.3.1.2	Requirements .....	320
17.3.1.3	Specifications.....	320
17.3.1.4	C Library.....	321
17.3.2	Other conventions.....	321
17.3.2.1	Type descriptions.....	321
17.3.2.1.1	Enumerated types.....	322
17.3.2.1.2	Bitmask types.....	322
17.3.2.1.3	Character sequences.....	323
17.3.2.1.3.1	Byte strings .....	323
17.3.2.1.3.2	Multibyte strings.....	324
17.3.2.1.3.3	Wide-character sequences.....	324
17.3.2.2	Functions within classes .....	324
17.3.2.3	Private members .....	324

iTeH STANDARD PREVIEW  
(standards.iteh.ai)

<https://standards.iteh.ai/catalog/standards/sist/55764e20-240a-47e6-ae0-59dc13467ca9/iso-iec-14882-2003>

17.4	Library-wide requirements .....	324
17.4.1	Library contents and organization .....	325
17.4.1.1	Library contents .....	325
17.4.1.2	Headers .....	325
17.4.1.3	Freestanding implementations .....	326
17.4.2	Using the library .....	326
17.4.2.1	Headers .....	326
17.4.2.2	Linkage .....	327
17.4.3	Constraints on programs .....	327
17.4.3.1	Reserved names .....	327
17.4.3.1.1	Macro names .....	327
17.4.3.1.2	Global names .....	327
17.4.3.1.3	External linkage .....	328
17.4.3.1.4	Types .....	328
17.4.3.2	Headers .....	328
17.4.3.3	Derived classes .....	328
17.4.3.4	Replacement functions .....	328
17.4.3.5	Handler functions .....	329
17.4.3.6	Other functions .....	329
17.4.3.7	Function arguments .....	330
17.4.3.8	Required paragraph .....	330
17.4.4	Conforming implementations .....	330
17.4.4.1	Headers .....	330
17.4.4.2	Restrictions on macro definitions .....	330
17.4.4.3	Global or non-member functions .....	330
17.4.4.4	Member functions .....	331
17.4.4.5	Reentrancy .....	331
17.4.4.6	Protection within classes .....	331
17.4.4.7	Derived classes .....	331
17.4.4.8	Restrictions on exception handling .....	331
18	Language support library .....	333
18.1	Types .....	333
18.2	Implementation properties .....	334
18.2.1	Numeric limits .....	334
18.2.1.1	Class template <code>numeric_limits</code> .....	334
18.2.1.2	<code>numeric_limits</code> members .....	335
18.2.1.3	Type <code>float_round_style</code> .....	339
18.2.1.4	Type <code>float_denorm_style</code> .....	340
18.2.1.5	<code>numeric_limits</code> specializations .....	340
18.2.2	C Library .....	341
18.3	Start and termination .....	342
18.4	Dynamic memory management .....	343
18.4.1	Storage allocation and deallocation .....	343
18.4.1.1	Single-object forms .....	343
18.4.1.2	Array forms .....	345
18.4.1.3	Placement forms .....	345
18.4.2	Storage allocation errors .....	346
18.4.2.1	Class <code>bad_alloc</code> .....	346
18.4.2.2	Type <code>new_handler</code> .....	347

18.4.2.3	set_new_handler .....	347
18.5	Type identification.....	347
18.5.1	Class type_info .....	347
18.5.2	Class bad_cast .....	348
18.5.3	Class bad_typeid.....	349
18.6	Exception handling .....	349
18.6.1	Class exception.....	349
18.6.2	Violating <i>exception-specifications</i> .....	350
18.6.2.1	Class bad_exception .....	350
18.6.2.2	Type unexpected_handler.....	351
18.6.2.3	set_unexpected.....	351
18.6.2.4	unexpected.....	351
18.6.3	Abnormal termination.....	351
18.6.3.1	Type terminate_handler .....	351
18.6.3.2	set_terminate.....	352
18.6.3.3	terminate.....	352
18.6.4	uncaught_exception.....	352
18.7	Other runtime support.....	352
19	Diagnostics library.....	355
19.1	Exception classes .....	355
19.1.1	Class logic_error .....	355
19.1.2	Class domain_error .....	356
19.1.3	Class invalid_argument .....	356
19.1.4	Class length_error.....	356
19.1.5	Class out_of_range.....	357
19.1.6	Class runtime_error .....	357
19.1.7	Class range_error .....	357
19.1.8	Class overflow_error.....	357
19.1.9	Class underflow_error.....	358
19.2	Assertions .....	358
19.3	Error numbers .....	358
20	General utilities library .....	359
20.1	Requirements .....	359
20.1.1	Equality comparison .....	359
20.1.2	Less than comparison .....	359
20.1.3	Copy construction.....	360
20.1.4	Default construction.....	360
20.1.5	Allocator requirements .....	360
20.2	Utility components.....	363
20.2.1	Operators.....	364
20.2.2	Pairs .....	364
20.3	Function objects.....	365
20.3.1	Base.....	367

iTech STANDARD PREVIEW  
(standards.iteh.ai)

20.3.2	Arithmetic operations .....	367
20.3.3	Comparisons .....	368
20.3.4	Logical operations .....	369
20.3.5	Negators .....	369
20.3.6	Binders .....	370
20.3.6.1	Class template binder1st .....	370
20.3.6.2	bind1st .....	370
20.3.6.3	Class template binder2nd .....	370
20.3.6.4	bind2nd .....	371
20.3.7	Adaptors for pointers to functions .....	371
20.3.8	Adaptors for pointers to members .....	372
20.4	Memory .....	374
20.4.1	The default allocator .....	374
20.4.1.1	allocator members .....	375
20.4.1.2	allocator globals .....	376
20.4.2	Raw storage iterator .....	376
20.4.3	Temporary buffers .....	377
20.4.4	Specialized algorithms .....	377
20.4.4.1	uninitialized_copy .....	377
20.4.4.2	uninitialized_fill .....	378
20.4.4.3	uninitialized_fill_n .....	378
20.4.5	Class template auto_ptr .....	378
20.4.5.1	auto_ptr constructors .....	379
20.4.5.2	auto_ptr members .....	379
20.4.5.3	auto_ptr conversions .....	380
20.4.6	Library .....	380
20.5	Date and time .....	381
21	Strings library .....	383
21.1	Character traits .....	383
21.1.1	Character traits requirements .....	383
21.1.2	traits typedefs .....	385
21.1.3	char_traits specializations .....	385
21.1.3.1	struct char_traits<char> .....	385
21.1.3.2	struct char_traits<wchar_t> .....	386
21.2	String classes .....	387
21.3	Class template basic_string .....	389
21.3.1	basic_string constructors .....	393
21.3.2	basic_string iterator support .....	396
21.3.3	basic_string capacity .....	396
21.3.4	basic_string element access .....	398
21.3.5	basic_string modifiers .....	398
21.3.5.1	basic_string::operator+= .....	398
21.3.5.2	basic_string::append .....	398
21.3.5.3	basic_string::assign .....	399
21.3.5.4	basic_string::insert .....	400
21.3.5.5	basic_string::erase .....	401
21.3.5.6	basic_string::replace .....	401
21.3.5.7	basic_string::copy .....	402