

---

---

## Programming languages — C++

*Langages de programmation — C++*

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

[ISO/IEC 14882:1998](https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b8f9-e7ce61967dd5/iso-iec-14882-1998)

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b8f9-e7ce61967dd5/iso-iec-14882-1998>

# Contents

1	General .....	1
1.1	Scope .....	1
1.2	Normative references .....	1
1.3	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 .....	
1.9	Program execution .....	5

© ISO/IEC 1998

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 the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

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 .....	33

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

ISO/IEC 14882:1998

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8d9-e7ee61967dd5/iso-iec-14882-1998>

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8d9-e7ee61967dd5/iso-iec-14882-1998>

3.4.3.1	Class members .....	35
3.4.3.2	Namespace members .....	35
3.4.4	Elaborated type specifiers .....	38
3.4.5	Class member access .....	39
3.4.6	Using-directives and namespace aliases .....	40
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.....	45
3.7.1	Static storage duration .....	46
3.7.2	Automatic storage duration.....	46
3.7.3	Dynamic storage duration.....	46
3.7.3.1	Allocation functions.....	47
3.7.3.2	Deallocation functions .....	47
3.7.4	Duration of sub-objects.....	48
3.8	Object Lifetime .....	48
3.9	Types.....	51
3.9.1	Fundamental types .....	53
3.9.2	Compound types .....	54
3.9.3	CV-qualifiers .....	55
3.10	Lvalues and rvalues.....	55
4	Standard conversions .....	57
4.1	Lvalue-to-rvalue conversion .....	57
4.2	Array-to-pointer conversion .....	58
4.3	Function-to-pointer conversion .....	58
4.4	Qualification conversions .....	58
4.5	Integral promotions.....	59
4.6	Floating point promotion .....	59
4.7	Integral conversions.....	60
4.8	Floating point conversions.....	60
4.9	Floating-integral conversions .....	60
4.10	Pointer conversions.....	60
4.11	Pointer to member conversions .....	61

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

ISO/IEC 14882:1998

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8f9-e7ce61967dd5/iso-iec-14882-1998>

4.12	Boolean conversions .....	61
5	Expressions .....	63
5.1	Primary expressions .....	64
5.2	Postfix expressions .....	66
5.2.1	Subscripting .....	66
5.2.2	Function call .....	66
5.2.3	Explicit type conversion (functional notation) .....	68
5.2.4	Pseudo destructor call .....	68
5.2.5	Class member access .....	68
5.2.6	Increment and decrement .....	69
5.2.7	Dynamic cast .....	70
5.2.8	Type identification .....	71
5.2.9	Static cast .....	72
5.2.10	Reinterpret cast .....	73
5.2.11	Const cast .....	74
5.3	Unary expressions .....	76
5.3.1	Unary operators .....	76
5.3.2	Increment and decrement .....	77
5.3.3	Sizeof .....	77
5.3.4	New .....	78
5.3.5	Delete .....	81
5.4	Explicit type conversion (cast notation) .....	82
5.5	Pointer-to-member operators .....	83
5.6	Multiplicative operators .....	83
5.7	Additive operators .....	84
5.8	Shift operators .....	85
5.9	Relational operators .....	85
5.10	Equality operators .....	86
5.11	Bitwise AND operator .....	87
5.12	Bitwise exclusive OR operator .....	87
5.13	Bitwise inclusive OR operator .....	87
5.14	Logical AND operator .....	87
5.15	Logical OR operator .....	88
5.16	Conditional operator .....	88
5.17	Assignment operators .....	89

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

ISO/IEC 14882:1998

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8f9-e7ce61967dd5/iso-iec-14882-1998>

5.18	Comma operator .....	90
5.19	Constant expressions .....	90
6	Statements .....	93
6.1	Labeled statement .....	93
6.2	Expression statement .....	93
6.3	Compound statement or block .....	93
6.4	Selection statements .....	94
6.4.1	The if statement .....	95
6.4.2	The switch statement .....	95
6.5	Iteration statements .....	95
6.5.1	The while statement .....	96
6.5.2	The do statement .....	96
6.5.3	The for statement .....	97
6.6	Jump statements .....	97
6.6.1	The break statement .....	97
6.6.2	The continue statement .....	98
6.6.3	The return statement .....	98
6.6.4	The goto statement .....	98
6.7	Declaration statement .....	98
6.8	Ambiguity resolution .....	99
7	Declarations .....	101
7.1	Specifiers .....	102
7.1.1	Storage class specifiers .....	103
7.1.2	Function specifiers .....	104
7.1.3	The typedef specifier .....	105
7.1.4	The friend specifier .....	106
7.1.5	Type specifiers .....	106
7.1.5.1	The <i>cv-qualifiers</i> .....	107
7.1.5.2	Simple type specifiers .....	108
7.1.5.3	Elaborated type specifiers .....	109
7.2	Enumeration declarations .....	110
7.3	Namespaces .....	112
7.3.1	Namespace definition .....	112
7.3.1.1	Unnamed namespaces .....	113
7.3.1.2	Namespace member definitions .....	113
7.3.2	Namespace alias .....	115
7.3.3	The using declaration .....	115
7.3.4	Using directive .....	120
7.4	The asm declaration .....	123

7.5	Linkage specifications .....	123
8	Declarators .....	127
8.1	Type names .....	128
8.2	Ambiguity resolution .....	128
8.3	Meaning of declarators .....	130
8.3.1	Pointers .....	131
8.3.2	References .....	132
8.3.3	Pointers to members .....	133
8.3.4	Arrays .....	133
8.3.5	Functions .....	135
8.3.6	Default arguments .....	137
8.4	Function definitions .....	140
8.5	Initializers .....	141
8.5.1	Aggregates .....	144
8.5.2	Character arrays .....	146
8.5.3	References .....	147
9	Classes .....	149
9.1	Class names .....	149
9.2	Class members .....	151
9.3	Member functions .....	153
9.3.1	Nonstatic member functions .....	154
9.3.2	The <code>this</code> pointer .....	155
9.4	Static members .....	156
9.4.1	Static member functions .....	157
9.4.2	Static data members .....	157
9.5	Unions .....	158
9.6	Bit-fields .....	159
9.7	Nested class declarations .....	160
9.8	Local class declarations .....	161
9.9	Nested type names .....	161
10	Derived classes .....	163
10.1	Multiple base classes .....	164
10.2	Member name lookup .....	165
10.3	Virtual functions .....	168

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

[ISO/IEC 14882:1998](https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b89-e7ee61967dd5/iso-iec-14882-1998)

[https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b89-](https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b89-e7ee61967dd5/iso-iec-14882-1998)

[e7ee61967dd5/iso-iec-14882-1998](https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43c8-b89-e7ee61967dd5/iso-iec-14882-1998)

10.4	Abstract classes .....	172
11	Member access control .....	175
11.1	Access specifiers.....	176
11.2	Accessibility of base classes and base class members.....	177
11.3	Access declarations.....	178
11.4	Friends .....	179
11.5	Protected member access .....	182
11.6	Access to virtual functions.....	183
11.7	Multiple access .....	183
11.8	Nested classes .....	184
12	Special member functions.....	185
12.1	Constructors.....	185
12.2	Temporary objects .....	187
12.3	Conversions .....	188
12.3.1	Conversion by constructor.....	189
12.3.2	Conversion functions.....	190
12.4	Destructors .....	191
12.5	Free store .....	194
12.6	Initialization.....	195
12.6.1	Explicit initialization .....	196
12.6.2	Initializing bases and members.....	197
12.7	Construction and destruction .....	200
12.8	Copying class objects .....	203
13	Overloading .....	209
13.1	Overloadable declarations.....	209
13.2	Declaration matching.....	211
13.3	Overload resolution .....	212
13.3.1	Candidate functions and argument lists.....	213
13.3.1.1	Function call syntax.....	214
13.3.1.1.1	Call to named function.....	214
13.3.1.1.2	Call to object of class type.....	215
13.3.1.2	Operators in expressions.....	216

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

[ISO/IEC 14882:1998](https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8f9-e7ce61967dd5/iso-iec-14882-1998)

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8f9-e7ce61967dd5/iso-iec-14882-1998>



13.3.1.3	Initialization by constructor .....	218
13.3.1.4	Copy-initialization of class by user-defined conversion.....	218
13.3.1.5	Initialization by conversion function .....	218
13.3.1.6	Initialization by conversion function for direct reference binding .....	219
13.3.2	Viable functions.....	219
13.3.3	Best Viable Function .....	219
13.3.3.1	Implicit conversion sequences .....	221
13.3.3.1.1	Standard conversion sequences .....	222
13.3.3.1.2	User-defined conversion sequences .....	223
13.3.3.1.3	Ellipsis conversion sequences.....	223
13.3.3.1.4	Reference binding .....	224
13.3.3.2	Ranking implicit conversion sequences.....	224
13.4	Address of overloaded function .....	226
13.5	Overloaded operators .....	227
13.5.1	Unary operators.....	228
13.5.2	Binary operators.....	229
13.5.3	Assignment .....	229
13.5.4	Function call .....	229
13.5.5	Subscripting .....	230
13.5.6	Class member access .....	230
13.5.7	Increment and decrement.....	230
13.6	Built-in operators .....	231
14	Templates.....	235
14.1	Template parameters.....	236
14.2	Names of template specializations.....	238
14.3	Template arguments.....	239
14.3.1	Template type arguments.....	241
14.3.2	Template non-type arguments .....	242
14.3.3	Template template arguments.....	243
14.4	Type equivalence .....	244
14.5	Template declarations .....	244
14.5.1	Class templates .....	244
14.5.1.1	Member functions of class templates.....	245
14.5.1.2	Member classes of class templates .....	246
14.5.1.3	Static data members of class templates.....	246
14.5.2	Member templates .....	246
14.5.3	Friends .....	248
14.5.4	Class template partial specializations .....	250
14.5.4.1	Matching of class template partial specializations .....	252
14.5.4.2	Partial ordering of class template specializations .....	252
14.5.4.3	Members of class template specializations .....	253
14.5.5	Function templates.....	254
14.5.5.1	Function template overloading .....	254
14.5.5.2	Partial ordering of function templates .....	256

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

ISO/IEC 14882:1998

<https://standards.iteh.ai/catalog/standards/sist/85a64a6d-6a7b-43e8-b8f9-e7ce61967dd5/iso-iec-14882-1998>

14.6	Name resolution.....	257
14.6.1	Locally declared names .....	260
14.6.2	Dependent names.....	262
14.6.2.1	Dependent types.....	263
14.6.2.2	Type-dependent expressions.....	263
14.6.2.3	Value-dependent expressions .....	264
14.6.2.4	Dependent template arguments.....	264
14.6.3	Non-dependent names.....	265
14.6.4	Dependent name resolution .....	265
14.6.4.1	Point of instantiation.....	265
14.6.4.2	Candidate functions .....	266
14.6.5	Friend names declared within a class template.....	266
14.7	Template instantiation and specialization.....	267
14.7.1	Implicit instantiation.....	268
14.7.2	Explicit instantiation.....	271
14.7.3	Explicit specialization.....	272
14.8	Function template specializations.....	277
14.8.1	Explicit template argument specification .....	278
14.8.2	Template argument deduction .....	280
14.8.2.1	Deducing template arguments from a function call.....	282
14.8.2.2	Deducing template arguments taking the address of a function template .....	283
14.8.2.3	Deducing conversion function template arguments.....	283
14.8.2.4	Deducing template arguments from a type .....	283
14.8.3	Overload resolution .....	288
15	Exception handling .....	291
15.1	Throwing an exception .....	292
15.2	Constructors and destructors.....	294
15.3	Handling an exception .....	294
15.4	Exception specifications .....	296
15.5	Special functions.....	298
15.5.1	The <code>terminate()</code> function .....	298
15.5.2	The <code>unexpected()</code> function.....	299
15.5.3	The <code>uncaught_exception()</code> function.....	299
15.6	Exceptions and access.....	299
16	Preprocessing directives .....	301
16.1	Conditional inclusion.....	302
16.2	Source file inclusion .....	303
16.3	Macro replacement .....	304
16.3.1	Argument substitution .....	305
16.3.2	The <code>#</code> operator .....	305
16.3.3	The <code>##</code> operator .....	306

16.3.4	Rescanning and further replacement.....	306
16.3.5	Scope of macro definitions .....	306
16.4	Line control.....	308
16.5	Error directive .....	308
16.6	Pragma directive .....	308
16.7	Null directive .....	308
16.8	Predefined macro names.....	309
17	Library introduction.....	311
17.1	Definitions .....	311
17.1.1	arbitrary-positional stream.....	311
17.1.2	character.....	311
17.1.3	character container type .....	311
17.1.4	comparison function .....	311
17.1.5	component.....	312
17.1.6	default behavior .....	312
17.1.7	handler function.....	312
17.1.8	iostream class templates .....	312
17.1.9	modifier function .....	312
17.1.10	object state .....	312
17.1.11	narrow-oriented iostream classes.....	312
17.1.12	NTCTS.....	312
17.1.13	observer function.....	312
17.1.14	replacement function.....	312
17.1.15	required behavior .....	312
17.1.16	repositional stream.....	313
17.1.17	reserved function.....	313
17.1.18	traits class.....	313
17.1.19	wide-oriented iostream classes .....	313
17.2	Additional definitions .....	313
17.3	Method of description (Informative) .....	313
17.3.1	Structure of each subclause.....	313
17.3.1.1	Summary.....	314
17.3.1.2	Requirements .....	314
17.3.1.3	Specifications.....	314
17.3.1.4	C Library.....	315
17.3.2	Other conventions .....	315
17.3.2.1	Type descriptions.....	315
17.3.2.1.1	Enumerated types.....	316
17.3.2.1.2	Bitmask types.....	316
17.3.2.1.3	Character sequences.....	317
17.3.2.1.3.1	Byte strings .....	317
17.3.2.1.3.2	Multibyte strings.....	318
17.3.2.1.3.3	Wide-character sequences.....	318
17.3.2.2	Functions within classes .....	318
17.3.2.3	Private members .....	318

17.4	Library-wide requirements .....	318
17.4.1	Library contents and organization .....	319
17.4.1.1	Library contents .....	319
17.4.1.2	Headers .....	319
17.4.1.3	Freestanding implementations .....	320
17.4.2	Using the library .....	320
17.4.2.1	Headers .....	320
17.4.2.2	Linkage .....	321
17.4.3	Constraints on programs .....	321
17.4.3.1	Reserved names .....	321
17.4.3.1.1	Macro names .....	321
17.4.3.1.2	Global names .....	321
17.4.3.1.3	External linkage .....	322
17.4.3.1.4	Types .....	322
17.4.3.2	Headers .....	322
17.4.3.3	Derived classes .....	322
17.4.3.4	Replacement functions .....	322
17.4.3.5	Handler functions .....	323
17.4.3.6	Other functions .....	323
17.4.3.7	Function arguments .....	324
17.4.3.8	Required paragraph .....	324
17.4.4	Conforming implementations .....	324
17.4.4.1	Headers .....	324
17.4.4.2	Restrictions on macro definitions .....	324
17.4.4.3	Global functions .....	324
17.4.4.4	Member functions .....	325
17.4.4.5	Reentrancy .....	325
17.4.4.6	Protection within classes .....	325
17.4.4.7	Derived classes .....	325
17.4.4.8	Restrictions on exception handling .....	325
18	Language support library .....	327
18.1	Types .....	327
18.2	Implementation properties .....	328
18.2.1	Numeric limits .....	328
18.2.1.1	Template class <code>numeric_limits</code> .....	328
18.2.1.2	<code>numeric_limits</code> members .....	329
18.2.1.3	Type <code>float_round_style</code> .....	333
18.2.1.4	Type <code>float_denorm_style</code> .....	334
18.2.1.5	<code>numeric_limits</code> specializations .....	334
18.2.2	C Library .....	335
18.3	Start and termination .....	336
18.4	Dynamic memory management .....	337
18.4.1	Storage allocation and deallocation .....	337
18.4.1.1	Single-object forms .....	337
18.4.1.2	Array forms .....	338
18.4.1.3	Placement forms .....	339
18.4.2	Storage allocation errors .....	340
18.4.2.1	Class <code>bad_alloc</code> .....	340
18.4.2.2	Type <code>new_handler</code> .....	340

18.4.2.3	set_new_handler .....	341
18.5	Type identification.....	341
18.5.1	Class type_info.....	341
18.5.2	Class bad_cast .....	342
18.5.3	Class bad_typeid.....	342
18.6	Exception handling .....	343
18.6.1	Class exception.....	343
18.6.2	Violating <i>exception-specifications</i> .....	344
18.6.2.1	Class bad_exception .....	344
18.6.2.2	Type unexpected_handler.....	345
18.6.2.3	set_unexpected.....	345
18.6.2.4	unexpected .....	345
18.6.3	Abnormal termination.....	345
18.6.3.1	Type terminate_handler .....	345
18.6.3.2	set_terminate.....	345
18.6.3.3	terminate.....	345
18.6.4	uncaught_exception.....	346
18.7	Other runtime support.....	346
19	Diagnostics library.....	349
19.1	Exception classes .....	349
19.1.1	Class logic_error .....	349
19.1.2	Class domain_error.....	350
19.1.3	Class invalid_argument .....	350
19.1.4	Class length_error.....	350
19.1.5	Class out_of_range.....	351
19.1.6	Class runtime_error .....	351
19.1.7	Class range_error .....	351
19.1.8	Class overflow_error.....	351
19.1.9	Class underflow_error.....	352
19.2	Assertions .....	352
19.3	Error numbers .....	352
20	General utilities library .....	353
20.1	Requirements .....	353
20.1.1	Equality comparison .....	353
20.1.2	Less than comparison .....	353
20.1.3	Copy construction.....	354
20.1.4	Default construction.....	354
20.1.5	Allocator requirements .....	354
20.2	Utility components.....	357
20.2.1	Operators.....	357
20.2.2	Pairs .....	358
20.3	Function objects.....	359
20.3.1	Base.....	361

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

20.3.2	Arithmetic operations .....	361
20.3.3	Comparisons .....	362
20.3.4	Logical operations .....	363
20.3.5	Negators .....	363
20.3.6	Binders .....	364
20.3.6.1	Template class binder1st .....	364
20.3.6.2	binder1st .....	364
20.3.6.3	Template class binder2nd .....	364
20.3.6.4	binder2nd .....	365
20.3.7	Adaptors for pointers to functions .....	365
20.3.8	Adaptors for pointers to members .....	366
20.4	Memory .....	368
20.4.1	The default allocator .....	368
20.4.1.1	allocator members .....	369
20.4.1.2	allocator globals .....	370
20.4.2	Raw storage iterator .....	370
20.4.3	Temporary buffers .....	371
20.4.4	Specialized algorithms .....	371
20.4.4.1	uninitialized_copy .....	371
20.4.4.2	uninitialized_fill .....	372
20.4.4.3	uninitialized_fill_n .....	372
20.4.5	Template class auto_ptr .....	372
20.4.5.1	auto_ptr constructors .....	373
20.4.5.2	auto_ptr members .....	373
20.4.5.3	auto_ptr conversions .....	374
20.4.6	C Library .....	374
20.5	Date and time .....	375
21	Strings library .....	377
21.1	Character traits .....	377
21.1.1	Character traits requirements .....	377
21.1.2	traits typedefs .....	379
21.1.3	char_traits specializations .....	379
21.1.3.1	struct char_traits<char> .....	379
21.1.3.2	struct char_traits<wchar_t> .....	380
21.2	String classes .....	381
21.3	Template class basic_string .....	383
21.3.1	basic_string constructors .....	387
21.3.2	basic_string iterator support .....	390
21.3.3	basic_string capacity .....	390
21.3.4	basic_string element access .....	391
21.3.5	basic_string modifiers .....	392
21.3.5.1	basic_string::operator+= .....	392
21.3.5.2	basic_string::append .....	392
21.3.5.3	basic_string::assign .....	393
21.3.5.4	basic_string::insert .....	393
21.3.5.5	basic_string::erase .....	394
21.3.5.6	basic_string::replace .....	395
21.3.5.7	basic_string::copy .....	396

21.3.5.8	<code>basic_string::swap</code> .....	397
21.3.6	<code>basic_string</code> string operations .....	397
21.3.6.1	<code>basic_string::find</code> .....	397
21.3.6.2	<code>basic_string::rfind</code> .....	398
21.3.6.3	<code>basic_string::find_first_of</code> .....	398
21.3.6.4	<code>basic_string::find_last_of</code> .....	399
21.3.6.5	<code>basic_string::find_first_not_of</code> .....	399
21.3.6.6	<code>basic_string::find_last_not_of</code> .....	400
21.3.6.7	<code>basic_string::substr</code> .....	400
21.3.6.8	<code>basic_string::compare</code> .....	400
21.3.7	<code>basic_string</code> non-member functions .....	401
21.3.7.1	<code>operator+</code> .....	401
21.3.7.2	<code>operator==</code> .....	40 2
21.3.7.3	<code>operator!=</code> .....	40 2
21.3.7.4	<code>operator&lt;</code> .....	403
21.3.7.5	<code>operator&gt;</code> .....	403
21.3.7.6	<code>operator&lt;=</code> .....	40 3
21.3.7.7	<code>operator&gt;=</code> .....	40 4
21.3.7.8	<code>swap</code> .....	404
21.3.7.9	Inserters and extractors .....	404
21.4	Null-terminated sequence utilities .....	405
22	Localization library .....	409
22.1	Locales .....	409
22.1.1	Class locale .....	410
22.1.1.1	locale types .....	412
22.1.1.1.1	Type <code>locale::category</code> .....	412
22.1.1.1.2	Class <code>locale::facet</code> .....	414
22.1.1.1.3	Class <code>locale::id</code> .....	414
22.1.1.2	locale constructors and destructor .....	415
22.1.1.3	locale members .....	416
22.1.1.4	locale operators .....	416
22.1.1.5	locale static members .....	416
22.1.2	locale globals .....	417
22.1.3	Convenience interfaces .....	417
22.1.3.1	Character classification .....	417
22.1.3.2	Character conversions .....	417
22.2	Standard locale categories .....	418
22.2.1	The <code>ctype</code> category .....	418
22.2.1.1	Template class <code>ctype</code> .....	418
22.2.1.1.1	<code>ctype</code> members .....	419
22.2.1.1.2	<code>ctype</code> virtual functions .....	420
22.2.1.2	Template class <code>ctype_byname</code> .....	421
22.2.1.3	<code>ctype</code> specializations .....	422
22.2.1.3.1	<code>ctype&lt;char&gt;</code> destructor .....	423
22.2.1.3.2	<code>ctype&lt;char&gt;</code> members .....	423
22.2.1.3.3	<code>ctype&lt;char&gt;</code> static members .....	424
22.2.1.3.4	<code>ctype&lt;char&gt;</code> virtual functions .....	424
22.2.1.4	Class <code>ctype_byname&lt;char&gt;</code> .....	425
22.2.1.5	Template class <code>codecvt</code> .....	425
22.2.1.5.1	<code>codecvt</code> members .....	426

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