

Contents

1	General	1-1
1.1	Scope	1-1
1.2	Normative references	1-1
1.3	Implementation compliance	1-2
1.4	Definitions	1-2
1.4.1	argument	1-2
1.4.2	diagnostic message	1-2
1.4.3	dynamic type	1-3
1.4.4	ill-formed program	1-3
1.4.5	implementation-defined behavior	1-3
1.4.6	implementation limits	1-3
1.4.7	locale-specific behavior	1-3
1.4.8	multibyte character	1-3
1.4.9	parameter	1-3
1.4.10	signature	1-3
1.4.11	static type	1-3
1.4.12	undefined behavior	1-3
1.4.13	unspecified behavior	1-4
1.4.14	well-formed program	1-4
1.5	Syntax notation	1-4
1.6	The C++ memory model	1-4
1.7	The C++ object model	1-4
1.8	Program execution	1-5
2	Lexical conventions	2-1
2.1	Phases of translation	2-1
2.2	Character sets	2-2
2.3	Trigraph sequences	2-3
2.4	Preprocessing tokens	2-3

2.5	Alternative tokens	2-4
2.6	Tokens	2-4
2.7	Comments	2-5
2.8	Header names	2-5
2.9	Preprocessing numbers.....	2-5
2.10	Identifiers	2-6
2.11	Keywords	2-6
2.12	Operators and punctuators.....	2-7
2.13	Literals.....	2-7
2.13.1	Integer literals.....	2-7
2.13.2	Character literals	2-8
2.13.3	Floating literals	2-10
2.13.4	String literals	2-11
2.13.5	Boolean literals.....	2-12
3	Basic concepts	3-1
3.1	Declarations and definitions.....	3-1
3.2	One definition rule	3-2
3.3	Declarative regions and scopes	3-4
3.3.1	Point of declaration	3-5
3.3.2	Local scope.....	3-6
3.3.3	Function prototype scope	3-6
3.3.4	Function scope	3-6
3.3.5	Namespace scope	3-6
3.3.6	Class scope	3-7
3.3.7	Name hiding	3-8
3.4	Name look up	3-9
3.4.1	Unqualified name look up.....	3-9
3.4.2	Argument-dependent name lookup	3-12
3.4.3	Qualified name look up.....	3-13
3.4.3.1	Class members	3-14
3.4.3.2	Namespace members.....	3-14
3.4.4	Elaborated type specifiers	3-18
3.4.5	Class member access.....	3-19
3.4.6	Using-directives and namespace aliases	3-20
3.5	Program and linkage	3-20
3.6	Start and termination	3-22
3.6.1	Main function.....	3-22
3.6.2	Initialization of non-local objects.....	3-23
3.6.3	Termination.....	3-24

3.7	Storage duration	3-24
3.7.1	Static storage duration.....	3-25
3.7.2	Automatic storage duration	3-25
3.7.3	Dynamic storage duration	3-25
3.7.3.1	Allocation functions.....	3-26
3.7.3.2	Deallocation functions	3-26
3.7.4	Duration of sub-objects	3-27
3.8	Object Lifetime	3-27
3.9	Types.....	3-30
3.9.1	Fundamental types	3-32
3.9.2	Compound types	3-33
3.9.3	CV-qualifiers.....	3-34
3.10	Lvalues and rvalues.....	3-34
4	Standard conversions	4-1
4.1	Lvalue-to-rvalue conversion	4-2
4.2	Array-to-pointer conversion.....	4-2
4.3	Function-to-pointer conversion.....	4-2
4.4	Qualification conversions.....	4-2
4.5	Integral promotions	4-3
4.6	Floating point promotion	4-4
4.7	Integral conversions	4-4
4.8	Floating point conversions	4-4
4.9	Floating-integral conversions.....	4-4
4.10	Pointer conversions	4-4
4.11	Pointer to member conversions.....	4-5
4.12	Boolean conversions	4-5
5	Expressions	5-1
5.1	Primary expressions	5-2
5.2	Postfix expressions.....	5-4
5.2.1	Subscripting	5-4
5.2.2	Function call.....	5-5
5.2.3	Explicit type conversion (functional notation).....	5-6
5.2.4	Pseudo destructor call	5-6
5.2.5	Class member access.....	5-6
5.2.6	Increment and decrement	5-8

5.2.7	Dynamic cast.....	5-8
5.2.8	Type identification	5-9
5.2.9	Static cast	5-10
5.2.10	Reinterpret cast.....	5-11
5.2.11	Const cast	5-12
5.3	Unary expressions	5-13
5.3.1	Unary operators	5-14
5.3.2	Increment and decrement	5-15
5.3.3	Sizeof.....	5-15
5.3.4	New	5-16
5.3.5	Delete	5-19
5.4	Explicit type conversion (cast notation).....	5-20
5.5	Pointer-to-member operators	5-21
5.6	Multiplicative operators	5-21
5.7	Additive operators	5-22
5.8	Shift operators	5-23
5.9	Relational operators	5-23
5.10	Equality operators	5-24
5.11	Bitwise AND operator	5-25
5.12	Bitwise exclusive OR operator	5-25
5.13	Bitwise inclusive OR operator.....	5-25
5.14	Logical AND operator	5-25
5.15	Logical OR operator	5-26
5.16	Conditional operator.....	5-26
5.17	Assignment operators.....	5-27
5.18	Comma operator	5-28
5.19	Constant expressions	5-28
6	Statements	6-1
6.1	Labeled statement.....	6-1
6.2	Expression statement.....	6-1
6.3	Compound statement or block	6-1
6.4	Selection statements	6-2

6.4.1	The <code>if</code> statement	6-3
6.4.2	The <code>switch</code> statement.....	6-3
6.5	Iteration statements	6-3
6.5.1	The <code>while</code> statement	6-4
6.5.2	The <code>do</code> statement	6-5
6.5.3	The <code>for</code> statement.....	6-5
6.6	Jump statements	6-5
6.6.1	The <code>break</code> statement	6-6
6.6.2	The <code>continue</code> statement.....	6-6
6.6.3	The <code>return</code> statement.....	6-6
6.6.4	The <code>goto</code> statement.....	6-6
6.7	Declaration statement.....	6-6
6.8	Ambiguity resolution	6-7
7	Declarations.....	7-1
7.1	Specifiers.....	7-2
7.1.1	Storage class specifiers	7-3
7.1.2	Function specifiers	7-4
7.1.3	The <code>typedef</code> specifier	7-5
7.1.4	The <code>friend</code> specifier	7-6
7.1.5	Type specifiers	7-6
7.1.5.1	The <i>cv-qualifiers</i>	7-7
7.1.5.2	Simple type specifiers	7-8
7.1.5.3	Elaborated type specifiers	7-9
7.2	Enumeration declarations.....	7-10
7.3	Namespaces.....	7-12
7.3.1	Namespace definition.....	7-12
7.3.1.1	Unnamed namespaces	7-13
7.3.1.2	Namespace member definitions	7-13
7.3.2	Namespace alias	7-15
7.3.3	The <code>using</code> declaration.....	7-15
7.3.4	Using directive	7-20
7.4	The <code>asm</code> declaration.....	7-23
7.5	Linkage specifications.....	7-23
8	Declarators	8-1
8.1	Type names	8-2
8.2	Ambiguity resolution	8-3
8.3	Meaning of declarators.....	8-4
8.3.1	Pointers.....	8-5
8.3.2	References	8-6
8.3.3	Pointers to members.....	8-7

8.3.4	Arrays.....	8-8
8.3.5	Functions.....	8-9
8.3.6	Default arguments	8-11
8.4	Function definitions	8-14
8.5	Initializers.....	8-15
8.5.1	Aggregates.....	8-18
8.5.2	Character arrays.....	8-20
8.5.3	References	8-21
9	Classes.....	9-1
9.1	Class names.....	9-2
9.2	Class members	9-3
9.3	Member functions	9-5
9.3.1	Nonstatic member functions	9-6
9.3.2	The <code>this</code> pointer.....	9-7
9.4	Static members.....	9-8
9.4.1	Static member functions.....	9-9
9.4.2	Static data members	9-9
9.5	Unions	9-10
9.6	Bit-fields.....	9-11
9.7	Nested class declarations.....	9-12
9.8	Local class declarations.....	9-13
9.9	Nested type names.....	9-13
10	Derived classes.....	10-1
10.1	Multiple base classes.....	10-2
10.2	Member name lookup	10-4
10.3	Virtual functions.....	10-6
10.4	Abstract classes	10-10
11	Member access control.....	11-1
11.1	Access specifiers	11-2
11.2	Accessibility of base classes and base class members	11-3
11.3	Access declarations	11-4
11.4	Friends.....	11-5

11.5	Protected member access	11-8
11.6	Access to virtual functions	11-9
11.7	Multiple access.....	11-9
11.8	Nested classes.....	11-10
12	Special member functions	12-1
12.1	Constructors	12-1
12.2	Temporary objects.....	12-3
12.3	Conversions.....	12-4
12.3.1	Conversion by constructor	12-5
12.3.2	Conversion functions	12-6
12.4	Destructors	12-7
12.5	Free store	12-9
12.6	Initialization	12-12
12.6.1	Explicit initialization.....	12-12
12.6.2	Initializing bases and members	12-13
12.7	Construction and destruction.....	12-16
12.8	Copying class objects.....	12-19
13	Overloading.....	13-1
13.1	Overloadable declarations	13-1
13.2	Declaration matching	13-3
13.3	Overload resolution.....	13-4
13.3.1	Candidate functions and argument lists	13-5
13.3.1.1	Function call syntax	13-6
13.3.1.1.1	Call to named function.....	13-7
13.3.1.1.2	Call to object of class type	13-7
13.3.1.2	Operators in expressions	13-8
13.3.1.3	Initialization by constructor	13-10
13.3.1.4	Copy-initialization of class by user-defined conversion	13-10
13.3.1.5	Initialization by conversion function	13-11
13.3.1.6	Initialization by conversion function for direct reference binding.....	13-11
13.3.2	Viable functions	13-11
13.3.3	Best Viable Function.....	13-12
13.3.3.1	Implicit conversion sequences	13-13
13.3.3.1.1	Standard conversion sequences.....	13-14
13.3.3.1.2	User-defined conversion sequences	13-15
13.3.3.1.3	Ellipsis conversion sequences	13-15
13.3.3.1.4	Reference binding	13-15
13.3.3.2	Ranking implicit conversion sequences	13-16

13.4	Address of overloaded function	13–18
13.5	Overloaded operators	13–19
13.5.1	Unary operators	13–20
13.5.2	Binary operators	13–21
13.5.3	Assignment.....	13–21
13.5.4	Function call.....	13–21
13.5.5	Subscripting	13–21
13.5.6	Class member access.....	13–22
13.5.7	Increment and decrement	13–22
13.6	Built-in operators	13–22
14	Templates	14–1
14.1	Template parameters	14–2
14.2	Names of template specializations	14–4
14.3	Template arguments	14–6
14.3.1	Template type arguments	14–7
14.3.2	Template non-type arguments.....	14–8
14.3.3	Template template arguments	14–10
14.4	Type equivalence	14–10
14.5	Template declarations	14–11
14.5.1	Class templates.....	14–11
14.5.1.1	Member functions of class templates	14–12
14.5.1.2	Member classes of class templates.....	14–12
14.5.1.3	Static data members of class templates.....	14–13
14.5.2	Member templates.....	14–13
14.5.3	Friends.....	14–15
14.5.4	Class template partial specializations.....	14–16
14.5.4.1	Matching of class template partial specializations.....	14–18
14.5.4.2	Partial ordering of class template specializations	14–19
14.5.4.3	Members of class template specializations	14–19
14.5.5	Function templates	14–21
14.5.5.1	Function template overloading.....	14–21
14.5.5.2	Partial ordering of function templates.....	14–22
14.6	Name resolution	14–24
14.6.1	Locally declared names.....	14–26
14.6.2	Dependent names	14–29
14.6.2.1	Dependent types	14–30
14.6.2.2	Type-dependent expressions	14–30
14.6.2.3	Value-dependent expressions.....	14–31
14.6.2.4	Dependent template arguments	14–31
14.6.3	Non-dependent names	14–32
14.6.4	Dependent name resolution.....	14–32
14.6.4.1	Point of instantiation	14–32
14.6.4.2	Candidate functions.....	14–33
14.6.5	Friend names declared within a class template	14–33

14.7	Template instantiation and specialization	14-34
14.7.1	Implicit instantiation	14-35
14.7.2	Explicit instantiation	14-38
14.7.3	Explicit specialization	14-39
14.8	Function template specializations	14-45
14.8.1	Explicit template argument specification	14-45
14.8.2	Template argument deduction	14-47
14.8.2.1	Deducing template arguments from a function call	14-47
14.8.2.2	Deducing template arguments taking the address of a function template	14-48
14.8.2.3	Deducing conversion function template arguments	14-48
14.8.2.4	Deducing template arguments from a type	14-49
14.8.3	Overload resolution	14-53
15	Exception handling	15-1
15.1	Throwing an exception	15-2
15.2	Constructors and destructors	15-4
15.3	Handling an exception	15-4
15.4	Exception specifications	15-6
15.5	Special functions	15-9
15.5.1	The <code>terminate()</code> function	15-9
15.5.2	The <code>unexpected()</code> function	15-9
15.5.3	The <code>uncaught_exception()</code> function	15-10
15.6	Exceptions and access	15-10
16	Preprocessing directives	16-1
16.1	Conditional inclusion	16-2
16.2	Source file inclusion	16-3
16.3	Macro replacement	16-4
16.3.1	Argument substitution	16-5
16.3.2	The <code>#</code> operator	16-6
16.3.3	The <code>##</code> operator	16-6
16.3.4	Rescanning and further replacement	16-6
16.3.5	Scope of macro definitions	16-6
16.4	Line control	16-8
16.5	Error directive	16-8
16.6	Pragma directive	16-9
16.7	Null directive	16-9
16.8	Predefined macro names	16-9

17	Library introduction	17-1
17.1	Definitions.....	17-1
17.1.1	comparison function.....	17-1
17.1.2	component.....	17-1
17.1.3	default behavior.....	17-2
17.1.4	handler function	17-2
17.1.5	modifier function.....	17-2
17.1.6	object state.....	17-2
17.1.7	observer function.....	17-2
17.1.8	replacement function.....	17-2
17.1.9	required behavior	17-2
17.1.10	reserved function.....	17-2
17.2	Method of description (Informative).....	17-2
17.2.1	Structure of each subclause	17-2
17.2.1.1	Summary	17-3
17.2.1.2	Requirements.....	17-3
17.2.1.3	Specifications	17-4
17.2.1.4	C Library	17-5
17.2.2	Other conventions	17-5
17.2.2.1	Type descriptions	17-5
17.2.2.1.1	Enumerated types	17-5
17.2.2.1.2	Bitmask types.....	17-5
17.2.2.1.3	Character sequences	17-6
17.2.2.1.3.1	Byte strings	17-6
17.2.2.1.3.2	Multibyte strings	17-7
17.2.2.1.3.3	Wide-character sequences	17-7
17.2.2.2	Functions within classes.....	17-7
17.2.2.3	Private members.....	17-7
17.3	Library-wide requirements.....	17-8
17.3.1	Library contents and organization.....	17-8
17.3.1.1	Library contents	17-8
17.3.1.2	Headers.....	17-8
17.3.1.3	Freestanding implementations	17-9
17.3.2	Using the library.....	17-9
17.3.2.1	Headers.....	17-9
17.3.2.2	Linkage.....	17-10
17.3.3	Constraints on programs	17-10
17.3.3.1	Reserved names.....	17-10
17.3.3.1.1	Macro names	17-10
17.3.3.1.2	Global names.....	17-11
17.3.3.1.3	External linkage	17-11
17.3.3.1.4	Types.....	17-11
17.3.3.2	Headers.....	17-11
17.3.3.3	Derived classes.....	17-11
17.3.3.4	Replacement functions	17-11
17.3.3.5	Handler functions	17-12
17.3.3.6	Other functions.....	17-12
17.3.3.7	Function arguments.....	17-13
17.3.3.8	Required paragraph	17-13
17.3.4	Conforming implementations	17-13
17.3.4.1	Headers.....	17-13

17.3.4.2	Restrictions on macro definitions.....	17-13
17.3.4.3	Global functions.....	17-13
17.3.4.4	Member functions.....	17-14
17.3.4.5	Reentrancy.....	17-14
17.3.4.6	Protection within classes.....	17-14
17.3.4.7	Derived classes.....	17-14
17.3.4.8	Restrictions on exception handling.....	17-14
18	Language support library.....	18-1
18.1	Types.....	18-1
18.2	Implementation properties.....	18-2
18.2.1	Numeric limits.....	18-2
18.2.1.1	Template class <code>numeric_limits</code>	18-3
18.2.1.2	<code>numeric_limits</code> members.....	18-3
18.2.1.3	Type <code>float_round_style</code>	18-7
18.2.1.4	<code>numeric_limits</code> specializations.....	18-8
18.2.2	C Library.....	18-9
18.3	Start and termination.....	18-9
18.4	Dynamic memory management.....	18-10
18.4.1	Storage allocation and deallocation.....	18-11
18.4.1.1	Single-object forms.....	18-11
18.4.1.2	Array forms.....	18-12
18.4.1.3	Placement forms.....	18-13
18.4.2	Storage allocation errors.....	18-14
18.4.2.1	Class <code>bad_alloc</code>	18-14
18.4.2.2	Type <code>new_handler</code>	18-14
18.4.2.3	<code>set_new_handler</code>	18-14
18.5	Type identification.....	18-14
18.5.1	Class <code>type_info</code>	18-15
18.5.2	Class <code>bad_cast</code>	18-16
18.5.3	Class <code>bad_typeid</code>	18-16
18.6	Exception handling.....	18-17
18.6.1	Class <code>exception</code>	18-17
18.6.2	Violating <i>exception-specifications</i>	18-18
18.6.2.1	Class <code>bad_exception</code>	18-18
18.6.2.2	Type <code>unexpected_handler</code>	18-18
18.6.2.3	<code>set_unexpected</code>	18-19
18.6.2.4	<code>unexpected</code>	18-19
18.6.3	Abnormal termination.....	18-19
18.6.3.1	Type <code>terminate_handler</code>	18-19
18.6.3.2	<code>set_terminate</code>	18-19
18.6.3.3	<code>terminate</code>	18-19
18.6.4	<code>uncaught_exception</code>	18-19
18.7	Other runtime support.....	18-20
19	Diagnostics library.....	19-1

19.1	Exception classes	19-1
19.1.1	Class <code>logic_error</code>	19-1
19.1.2	Class <code>domain_error</code>	19-2
19.1.3	Class <code>invalid_argument</code>	19-2
19.1.4	Class <code>length_error</code>	19-2
19.1.5	Class <code>out_of_range</code>	19-3
19.1.6	Class <code>runtime_error</code>	19-3
19.1.7	Class <code>range_error</code>	19-3
19.1.8	Class <code>overflow_error</code>	19-4
19.1.9	Class <code>underflow_error</code>	19-4
19.2	Assertions.....	19-4
19.3	Error numbers.....	19-4
20	General utilities library	20-1
20.1	Requirements.....	20-1
20.1.1	Equality comparison.....	20-1
20.1.2	Less than comparison	20-1
20.1.3	Copy construction	20-2
20.1.4	Default construction	20-2
20.1.5	Allocator requirements	20-2
20.2	Utility components	20-5
20.2.1	Operators	20-5
20.2.2	Pairs.....	20-6
20.3	Function objects	20-7
20.3.1	Base	20-8
20.3.2	Arithmetic operations.....	20-9
20.3.3	Comparisons.....	20-9
20.3.4	Logical operations	20-10
20.3.5	Negators	20-11
20.3.6	Binders	20-11
20.3.6.1	Template class <code>binder1st</code>	20-11
20.3.6.2	<code>bind1st</code>	20-12
20.3.6.3	Template class <code>binder2nd</code>	20-12
20.3.6.4	<code>bind2nd</code>	20-12
20.3.7	Adaptors for pointers to functions.....	20-12
20.3.8	Adaptors for pointers to members.....	20-13
20.4	Memory	20-14
20.4.1	The default allocator	20-15
20.4.1.1	allocator members	20-16
20.4.1.2	allocator globals.....	20-16
20.4.2	Raw storage iterator	20-17
20.4.3	Temporary buffers.....	20-17
20.4.4	Specialized algorithms	20-18
20.4.4.1	<code>uninitialized_copy</code>	20-18
20.4.4.2	<code>uninitialized_fill</code>	20-18
20.4.4.3	<code>uninitialized_fill_n</code>	20-18
20.4.5	Template class <code>auto_ptr</code>	20-18
20.4.5.1	<code>auto_ptr</code> constructors	20-19

20.4.5.2	auto_ptr members.....	20-20
20.4.6	C Library.....	20-20
20.5	Date and time.....	20-21
21	Strings library.....	21-1
21.1	Character traits.....	21-1
21.1.1	Definitions.....	21-1
21.1.2	Character traits requirements.....	21-2
21.1.3	traits typedefs.....	21-4
21.1.4	char_traits specializations.....	21-5
21.1.4.1	struct char_traits<char>.....	21-5
21.1.4.2	struct char_traits<wchar_t>.....	21-6
21.2	String classes.....	21-7
21.3	Template class basic_string.....	21-9
21.3.1	basic_string constructors.....	21-12
21.3.2	basic_string iterator support.....	21-15
21.3.3	basic_string capacity.....	21-15
21.3.4	basic_string element access.....	21-16
21.3.5	basic_string modifiers.....	21-17
21.3.5.1	basic_string::operator+=.....	21-17
21.3.5.2	basic_string::append.....	21-17
21.3.5.3	basic_string::assign.....	21-18
21.3.5.4	basic_string::insert.....	21-18
21.3.5.5	basic_string::erase.....	21-19
21.3.5.6	basic_string::replace.....	21-20
21.3.5.7	basic_string::copy.....	21-21
21.3.5.8	basic_string::swap.....	21-21
21.3.6	basic_string string operations.....	21-21
21.3.6.1	basic_string::find.....	21-22
21.3.6.2	basic_string::rfind.....	21-22
21.3.6.3	basic_string::find_first_of.....	21-23
21.3.6.4	basic_string::find_last_of.....	21-23
21.3.6.5	basic_string::find_first_not_of.....	21-24
21.3.6.6	basic_string::find_last_not_of.....	21-24
21.3.6.7	basic_string::substr.....	21-25
21.3.6.8	basic_string::compare.....	21-25
21.3.7	basic_string non-member functions.....	21-26
21.3.7.1	operator+.....	21-26
21.3.7.2	operator==.....	21-26
21.3.7.3	operator!=.....	21-27
21.3.7.4	operator<.....	21-27
21.3.7.5	operator>.....	21-27
21.3.7.6	operator<=.....	21-28
21.3.7.7	operator>=.....	21-28
21.3.7.8	swap.....	21-28
21.3.7.9	Inserters and extractors.....	21-28
21.4	Null-terminated sequence utilities.....	21-30
22	Localization library.....	22-1

22.1	Locales	22-1
22.1.1	Class locale.....	22-2
22.1.1.1	locale types.....	22-4
22.1.1.1.1	Type locale::category.....	22-4
22.1.1.1.2	Class locale::facet.....	22-6
22.1.1.1.3	Class locale::id.....	22-7
22.1.1.2	locale constructors and destructor.....	22-7
22.1.1.3	locale members.....	22-8
22.1.1.4	locale operators.....	22-8
22.1.1.5	locale static members.....	22-9
22.1.2	locale globals.....	22-9
22.1.3	Convenience interfaces.....	22-9
22.1.3.1	Character classification.....	22-9
22.1.3.2	Character conversions.....	22-10
22.2	Standard locale categories.....	22-10
22.2.1	The ctype category.....	22-10
22.2.1.1	Template class ctype.....	22-10
22.2.1.1.1	ctype members.....	22-11
22.2.1.1.2	ctype virtual functions.....	22-12
22.2.1.2	Template class ctype_byname.....	22-13
22.2.1.3	ctype specializations.....	22-14
22.2.1.3.1	ctype<char> destructor.....	22-15
22.2.1.3.2	ctype<char> members.....	22-15
22.2.1.3.3	ctype<char> static members.....	22-16
22.2.1.3.4	ctype<char> virtual functions.....	22-16
22.2.1.4	Class ctype_byname<char>.....	22-17
22.2.1.5	Template class codecvt.....	22-17
22.2.1.5.1	codecvt members.....	22-18
22.2.1.5.2	codecvt virtual functions.....	22-19
22.2.1.6	Template class codecvt_byname.....	22-20
22.2.2	The numeric category.....	22-21
22.2.2.1	Template class num_get.....	22-21
22.2.2.1.1	num_get members.....	22-22
22.2.2.1.2	num_get virtual functions.....	22-23
22.2.2.2	Template class num_put.....	22-25
22.2.2.2.1	num_put members.....	22-26
22.2.2.2.2	num_put virtual functions.....	22-26
22.2.3	The numeric punctuation facet.....	22-29
22.2.3.1	Template class numpunct.....	22-29
22.2.3.1.1	numpunct members.....	22-30
22.2.3.1.2	numpunct virtual functions.....	22-31
22.2.3.2	Template class numpunct_byname.....	22-31
22.2.4	The collate category.....	22-31
22.2.4.1	Template class collate.....	22-31
22.2.4.1.1	collate members.....	22-32
22.2.4.1.2	collate virtual functions.....	22-32
22.2.4.2	Template class collate_byname.....	22-33
22.2.5	The time category.....	22-33
22.2.5.1	Template class time_get.....	22-33
22.2.5.1.1	time_get members.....	22-34
22.2.5.1.2	time_get virtual functions.....	22-35
22.2.5.2	Template class time_get_byname.....	22-36
22.2.5.3	Template class time_put.....	22-36

22.2.5.3.1	time_put members.....	22-36
22.2.5.3.2	time_put virtual functions.....	22-37
22.2.5.4	Template class time_put_byname.....	22-37
22.2.6	The monetary category.....	22-37
22.2.6.1	Template class money_get.....	22-37
22.2.6.1.1	money_get members.....	22-38
22.2.6.1.2	money_get virtual functions.....	22-38
22.2.6.2	Template class money_put.....	22-39
22.2.6.2.1	money_put members.....	22-40
22.2.6.2.2	money_put virtual functions.....	22-40
22.2.6.3	Template class moneypunct.....	22-41
22.2.6.3.1	moneypunct members.....	22-42
22.2.6.3.2	moneypunct virtual functions.....	22-42
22.2.6.4	Template class moneypunct_byname.....	22-43
22.2.7	The message retrieval category.....	22-43
22.2.7.1	Template class messages.....	22-43
22.2.7.1.1	messages members.....	22-44
22.2.7.1.2	messages virtual functions.....	22-44
22.2.7.2	Template class messages_byname.....	22-45
22.2.8	Program-defined facets.....	22-45
22.3	C Library Locales.....	22-48
23	Containers library.....	23-1
23.1	Container requirements.....	23-1
23.1.1	Sequences.....	23-4
23.1.2	Associative containers.....	23-6
23.2	Sequences.....	23-9
23.2.1	Template class deque.....	23-12
23.2.1.1	deque constructors, copy, and assignment.....	23-14
23.2.1.2	deque capacity.....	23-15
23.2.1.3	deque modifiers.....	23-15
23.2.1.4	deque specialized algorithms.....	23-16
23.2.2	Template class list.....	23-16
23.2.2.1	list constructors, copy, and assignment.....	23-18
23.2.2.2	list capacity.....	23-18
23.2.2.3	list modifiers.....	23-19
23.2.2.4	list operations.....	23-19
23.2.2.5	list specialized algorithms.....	23-21
23.2.3	Container adaptors.....	23-21
23.2.3.1	Template class queue.....	23-21
23.2.3.2	Template class priority_queue.....	23-22
23.2.3.2.1	priority_queue constructors.....	23-23
23.2.3.2.2	priority_queue members.....	23-23
23.2.3.3	Template class stack.....	23-23
23.2.4	Template class vector.....	23-24
23.2.4.1	vector constructors, copy, and assignment.....	23-26
23.2.4.2	vector capacity.....	23-26
23.2.4.3	vector modifiers.....	23-27
23.2.4.4	vector specialized algorithms.....	23-27
23.2.5	Class vector<bool>.....	23-27

23.3	Associative containers.....	23–29
23.3.1	Template class map.....	23–31
23.3.1.1	map constructors, copy, and assignment	23–34
23.3.1.2	map element access.....	23–34
23.3.1.3	map operations.....	23–34
23.3.1.4	map specialized algorithms.....	23–34
23.3.2	Template class multimap.....	23–35
23.3.2.1	multimap constructors	23–37
23.3.2.2	multimap operations.....	23–37
23.3.2.3	multimap specialized algorithms.....	23–38
23.3.3	Template class set.....	23–38
23.3.3.1	set constructors, copy, and assignment	23–40
23.3.3.2	set specialized algorithms.....	23–40
23.3.4	Template class multiset.....	23–40
23.3.4.1	multiset constructors	23–42
23.3.4.2	multiset specialized algorithms.....	23–43
23.3.5	Template class bitset	23–43
23.3.5.1	bitset constructors.....	23–45
23.3.5.2	bitset members.....	23–45
23.3.5.3	bitset operators	23–47
24	Iterators library.....	24–1
24.1	Iterator requirements	24–1
24.1.1	Input iterators	24–2
24.1.2	Output iterators.....	24–3
24.1.3	Forward iterators	24–4
24.1.4	Bidirectional iterators.....	24–5
24.1.5	Random access iterators	24–5
24.2	Header <iterator> synopsis	24–6
24.3	Iterator primitives.....	24–8
24.3.1	Iterator traits	24–8
24.3.2	Basic iterator	24–9
24.3.3	Standard iterator tags.....	24–9
24.3.4	Iterator operations	24–10
24.4	Predefined iterators	24–11
24.4.1	Reverse iterators.....	24–11
24.4.1.1	Template class reverse_iterator	24–11
24.4.1.2	reverse_iterator requirements	24–12
24.4.1.3	reverse_iterator operations.....	24–12
24.4.1.3.1	reverse_iterator constructor	24–12
24.4.1.3.2	Conversion	24–13
24.4.1.3.3	operator*	24–13
24.4.1.3.4	operator->.....	24–13
24.4.1.3.5	operator++.....	24–13
24.4.1.3.6	operator--	24–13
24.4.1.3.7	operator+	24–13
24.4.1.3.8	operator+=.....	24–14
24.4.1.3.9	operator-	24–14
24.4.1.3.10	operator-=.....	24–14
24.4.1.3.11	operator[].....	24–14

24.4.1.3.12	operator==.....	24-14
24.4.1.3.13	operator<.....	24-14
24.4.1.3.14	operator!=.....	24-14
24.4.1.3.15	operator>.....	24-14
24.4.1.3.16	operator>=.....	24-15
24.4.1.3.17	operator<=.....	24-15
24.4.1.3.18	operator-.....	24-15
24.4.1.3.19	operator+.....	24-15
24.4.2	Insert iterators.....	24-15
24.4.2.1	Template class back_insert_iterator.....	24-16
24.4.2.2	back_insert_iterator operations.....	24-16
24.4.2.2.1	back_insert_iterator constructor.....	24-16
24.4.2.2.2	back_insert_iterator::operator=.....	24-16
24.4.2.2.3	back_insert_iterator::operator*.....	24-16
24.4.2.2.4	back_insert_iterator::operator++.....	24-16
24.4.2.2.5	back_inserter.....	24-16
24.4.2.3	Template class front_insert_iterator.....	24-17
24.4.2.4	front_insert_iterator operations.....	24-17
24.4.2.4.1	front_insert_iterator constructor.....	24-17
24.4.2.4.2	front_insert_iterator::operator=.....	24-17
24.4.2.4.3	front_insert_iterator::operator*.....	24-17
24.4.2.4.4	front_insert_iterator::operator++.....	24-17
24.4.2.4.5	front_inserter.....	24-17
24.4.2.5	Template class insert_iterator.....	24-18
24.4.2.6	insert_iterator operations.....	24-18
24.4.2.6.1	insert_iterator constructor.....	24-18
24.4.2.6.2	insert_iterator::operator=.....	24-18
24.4.2.6.3	insert_iterator::operator*.....	24-18
24.4.2.6.4	insert_iterator::operator++.....	24-18
24.4.2.6.5	inserter.....	24-19
24.5	Stream iterators.....	24-19
24.5.1	Template class istream_iterator.....	24-19
24.5.1.1	istream_iterator constructors and destructor.....	24-20
24.5.1.2	istream_iterator operations.....	24-20
24.5.2	Template class ostream_iterator.....	24-21
24.5.2.1	ostream_iterator constructors and destructor.....	24-21
24.5.2.2	ostream_iterator operations.....	24-22
24.5.3	Template class istreambuf_iterator.....	24-22
24.5.3.1	Template class istreambuf_iterator::proxy.....	24-23
24.5.3.2	istreambuf_iterator constructors.....	24-23
24.5.3.3	istreambuf_iterator::operator*.....	24-23
24.5.3.4	istreambuf_iterator::operator++.....	24-24
24.5.3.5	istreambuf_iterator::equal.....	24-24
24.5.3.6	operator==.....	24-24
24.5.3.7	operator!=.....	24-24
24.5.4	Template class ostreambuf_iterator.....	24-24
24.5.4.1	ostreambuf_iterator constructors.....	24-25
24.5.4.2	ostreambuf_iterator operations.....	24-25
25	Algorithms library.....	25-1
25.1	Non-modifying sequence operations.....	25-9
25.1.1	For each.....	25-9

25.1.2	Find	25-10
25.1.3	Find End.....	25-10
25.1.4	Find First.....	25-10
25.1.5	Adjacent find.....	25-11
25.1.6	Count.....	25-11
25.1.7	Mismatch.....	25-11
25.1.8	Equal	25-11
25.1.9	Search.....	25-12
25.2	Mutating sequence operations.....	25-13
25.2.1	Copy	25-13
25.2.2	Swap.....	25-13
25.2.3	Transform.....	25-14
25.2.4	Replace.....	25-14
25.2.5	Fill	25-15
25.2.6	Generate	25-15
25.2.7	Remove	25-15
25.2.8	Unique.....	25-16
25.2.9	Reverse.....	25-16
25.2.10	Rotate	25-17
25.2.11	Random shuffle.....	25-17
25.2.12	Partitions	25-18
25.3	Sorting and related operations.....	25-18
25.3.1	Sorting.....	25-19
25.3.1.1	sort.....	25-19
25.3.1.2	stable_sort	25-19
25.3.1.3	partial_sort.....	25-19
25.3.1.4	partial_sort_copy.....	25-20
25.3.2	Nth element.....	25-20
25.3.3	Binary search.....	25-20
25.3.3.1	lower_bound	25-20
25.3.3.2	upper_bound	25-21
25.3.3.3	equal_range	25-21
25.3.3.4	binary_search.....	25-21
25.3.4	Merge	25-22
25.3.5	Set operations on sorted structures.....	25-22
25.3.5.1	includes.....	25-23
25.3.5.2	set_union.....	25-23
25.3.5.3	set_intersection	25-23
25.3.5.4	set_difference	25-24
25.3.5.5	set_symmetric_difference	25-24
25.3.6	Heap operations.....	25-24
25.3.6.1	push_heap.....	25-25
25.3.6.2	pop_heap.....	25-25
25.3.6.3	make_heap.....	25-25
25.3.6.4	sort_heap.....	25-25
25.3.7	Minimum and maximum.....	25-26
25.3.8	Lexicographical comparison	25-26
25.3.9	Permutation generators.....	25-27
25.4	C library algorithms	25-27
26	Numerics library.....	26-1

26.1	Numeric type requirements	26-1
26.2	Complex numbers	26-2
26.2.1	Header <complex> synopsis	26-2
26.2.2	Template class complex	26-3
26.2.3	complex specializations	26-4
26.2.4	complex member functions	26-5
26.2.5	complex member operators	26-6
26.2.6	complex non-member operations	26-6
26.2.7	complex value operations	26-8
26.2.8	complex transcendentals	26-8
26.3	Numeric arrays	26-9
26.3.1	Header <valarray> synopsis	26-9
26.3.2	Template class valarray	26-12
26.3.2.1	valarray constructors	26-14
26.3.2.2	valarray assignment	26-14
26.3.2.3	valarray element access	26-15
26.3.2.4	valarray subset operations	26-15
26.3.2.5	valarray unary operators	26-16
26.3.2.6	valarray computed assignment	26-16
26.3.2.7	valarray member functions	26-17
26.3.3	valarray non-member operations	26-18
26.3.3.1	valarray binary operators	26-18
26.3.3.2	valarray logical operators	26-19
26.3.3.3	valarray transcendentals	26-20
26.3.4	Class slice	26-21
26.3.4.1	slice constructors	26-21
26.3.4.2	slice access functions	26-21
26.3.5	Template class slice_array	26-21
26.3.5.1	slice_array constructors	26-22
26.3.5.2	slice_array assignment	26-22
26.3.5.3	slice_array computed assignment	26-23
26.3.5.4	slice_array fill function	26-23
26.3.6	The gslice class	26-23
26.3.6.1	gslice constructors	26-24
26.3.6.2	gslice access functions	26-24
26.3.7	Template class gslice_array	26-25
26.3.7.1	gslice_array constructors	26-25
26.3.7.2	gslice_array assignment	26-25
26.3.7.3	gslice_array computed assignment	26-26
26.3.7.4	gslice_array fill function	26-26
26.3.8	Template class mask_array	26-26
26.3.8.1	mask_array constructors	26-27
26.3.8.2	mask_array assignment	26-27
26.3.8.3	mask_array computed assignment	26-27
26.3.8.4	mask_array fill function	26-27
26.3.9	Template class indirect_array	26-27
26.3.9.1	indirect_array constructors	26-28
26.3.9.2	indirect_array assignment	26-28
26.3.9.3	indirect_array computed assignment	26-29
26.3.9.4	indirect_array fill function	26-29
26.4	Generalized numeric operations	26-29

26.4.1	Accumulate	26-30
26.4.2	Inner product	26-30
26.4.3	Partial sum.....	26-31
26.4.4	Adjacent difference	26-31
26.5	C Library	26-31
27	Input/output library	27-1
27.1	Iostreams requirements	27-1
27.1.1	Definitions.....	27-1
27.1.2	Limitations	27-2
27.1.2.1	Imbue Limitations.....	27-2
27.1.2.2	Positioning Type Limitations.....	27-2
27.2	Forward declarations	27-2
27.3	Standard iostream objects	27-5
27.3.1	Narrow stream objects.....	27-5
27.3.2	Wide stream objects	27-6
27.4	Iostreams base classes	27-6
27.4.1	Types.....	27-7
27.4.2	Class ios_base.....	27-7
27.4.2.1	Types.....	27-9
27.4.2.1.1	Class ios_base::failure	27-10
27.4.2.1.2	Type ios_base::fmtflags	27-10
27.4.2.1.3	Type ios_base::iostate.....	27-11
27.4.2.1.4	Type ios_base::openmode	27-11
27.4.2.1.5	Type ios_base::seekdir.....	27-11
27.4.2.1.6	Class ios_base::Init	27-12
27.4.2.2	ios_base fmtflags state functions.....	27-12
27.4.2.3	ios_base locale functions.....	27-13
27.4.2.4	ios_base static members	27-13
27.4.2.5	ios_base storage functions	27-14
27.4.2.6	ios_base callbacks.....	27-14
27.4.2.7	ios_base constructors/destructors	27-14
27.4.3	Template class fpos	27-15
27.4.3.1	fpos Constructor	27-15
27.4.3.2	fpos Members.....	27-15
27.4.4	fpos requirements	27-15
27.4.5	Template class basic_ios	27-16
27.4.5.1	basic_ios constructors.....	27-17
27.4.5.2	Member functions	27-18
27.4.5.3	basic_ios iostate flags functions	27-19
27.4.6	ios_base manipulators.....	27-20
27.4.6.1	fmtflags manipulators.....	27-20
27.4.6.2	adjustfield manipulators	27-21
27.4.6.3	basefield manipulators	27-21
27.4.6.4	floatfield manipulators.....	27-22
27.5	Stream buffers	27-22
27.5.1	Stream buffer requirements.....	27-22
27.5.2	Template class basic_streambuf<charT, traits>	27-23

27.5.2.1	<code>basic_streambuf</code> constructors	27-25
27.5.2.2	<code>basic_streambuf</code> public member functions.....	27-25
27.5.2.2.1	Locales	27-25
27.5.2.2.2	Buffer management and positioning	27-25
27.5.2.2.3	Get area	27-26
27.5.2.2.4	Putback.....	27-26
27.5.2.2.5	Put area.....	27-26
27.5.2.3	<code>basic_streambuf</code> protected member functions	27-27
27.5.2.3.1	Get area access	27-27
27.5.2.3.2	Put area access.....	27-27
27.5.2.4	<code>basic_streambuf</code> virtual functions	27-27
27.5.2.4.1	Locales	27-27
27.5.2.4.2	Buffer management and positioning	27-28
27.5.2.4.3	Get area	27-28
27.5.2.4.4	Putback.....	27-29
27.5.2.4.5	Put area.....	27-30
27.6	Formatting and manipulators	27-31
27.6.1	Input streams	27-31
27.6.1.1	Template class <code>basic_istream</code>	27-31
27.6.1.1.1	<code>basic_istream</code> constructors.....	27-33
27.6.1.1.2	Class <code>basic_istream::sentry</code>	27-34
27.6.1.2	Formatted input functions	27-35
27.6.1.2.1	Common requirements	27-35
27.6.1.2.2	Arithmetic Extractors	27-35
27.6.1.2.3	<code>basic_istream::operator>></code>	27-36
27.6.1.3	Unformatted input functions	27-37
27.6.1.4	Standard <code>basic_istream</code> manipulators	27-41
27.6.1.5	Template class <code>basic_iostream</code>	27-41
27.6.1.5.1	<code>basic_iostream</code> constructors.....	27-41
27.6.1.5.2	<code>basic_iostream</code> destructor	27-41
27.6.2	Output streams	27-42
27.6.2.1	Template class <code>basic_ostream</code>	27-42
27.6.2.2	<code>basic_ostream</code> constructors	27-43
27.6.2.3	Class <code>basic_ostream::sentry</code>	27-44
27.6.2.4	<code>basic_ostream</code> seek members.....	27-44
27.6.2.5	Formatted output functions	27-45
27.6.2.5.1	Common requirements	27-45
27.6.2.5.2	Arithmetic Inserters.....	27-45
27.6.2.5.3	<code>basic_ostream::operator<<</code>	27-46
27.6.2.5.4	Character inserter template functions.....	27-46
27.6.2.6	Unformatted output functions	27-47
27.6.2.7	Standard <code>basic_ostream</code> manipulators	27-48
27.6.3	Standard manipulators.....	27-48
27.7	String-based streams	27-50
27.7.1	Template class <code>basic_stringbuf</code>	27-51
27.7.1.1	<code>basic_stringbuf</code> constructors	27-52
27.7.1.2	Member functions	27-52
27.7.1.3	Overridden virtual functions	27-53
27.7.2	Template class <code>basic_istream</code>	27-55
27.7.2.1	<code>basic_istream</code> constructors.....	27-55
27.7.2.2	Member functions	27-56
27.7.3	Class <code>basic_ostream</code>	27-56

27.7.3.1	basic_ostringstream constructors.....	27-57
27.7.3.2	Member functions	27-57
27.7.4	Template class basic_stringstream.....	27-57
27.7.5	basic_stringstream constructors	27-58
27.7.6	Member functions	27-58
27.8	File-based streams	27-58
27.8.1	File streams	27-58
27.8.1.1	Template class basic_filebuf.....	27-59
27.8.1.2	basic_filebuf constructors	27-60
27.8.1.3	Member functions	27-61
27.8.1.4	Overridden virtual functions	27-62
27.8.1.5	Template class basic_ifstream	27-64
27.8.1.6	basic_ifstream constructors.....	27-65
27.8.1.7	Member functions	27-65
27.8.1.8	Template class basic_ofstream	27-66
27.8.1.9	basic_ofstream constructors.....	27-66
27.8.1.10	Member functions	27-66
27.8.1.11	Template class basic_fstream.....	27-67
27.8.1.12	basic_fstream constructors	27-67
27.8.1.13	Member functions	27-68
27.8.2	C Library files	27-68
A	Grammar summary.....	A-1
A.1	Keywords.....	A-1
A.2	Lexical conventions.....	A-1
A.3	Basic concepts	A-5
A.4	Expressions.....	A-5
A.5	Statements	A-8
A.6	Declarations.....	A-9
A.7	Declarators.....	A-11
A.8	Classes	A-13
A.9	Derived classes	A-14
A.10	Special member functions	A-14
A.11	Overloading	A-14
A.12	Templates	A-15
A.13	Exception handling.....	A-15
A.14	Preprocessing directives	A-16
B	Implementation quantities	B-1

C	Compatibility	C-1
C.1	C++ and ISO C.....	C-1
C.1.1	Clause 2: lexical conventions	C-1
C.1.2	Clause 3: basic concepts	C-2
C.1.3	Clause 5: expressions.....	C-4
C.1.4	Clause 6: statements	C-4
C.1.5	Clause 7: declarations	C-5
C.1.6	Clause 8: declarators.....	C-7
C.1.7	Clause 9: classes	C-8
C.1.8	Clause 12: special member functions	C-9
C.1.9	Clause 16: preprocessing directives.....	C-10
C.2	Standard C library.....	C-10
C.2.1	Modifications to headers.....	C-12
C.2.2	Modifications to definitions.....	C-12
C.2.2.1	Type <code>wchar_t</code>	C-12
C.2.2.2	Header <code><iso646.h></code>	C-12
C.2.2.3	Macro <code>NULL</code>	C-13
C.2.3	Modifications to declarations.....	C-13
C.2.4	Modifications to behavior.....	C-13
C.2.4.1	Macro <code>offsetof(type, member-designator)</code>	C-13
C.2.4.2	Memory allocation functions.....	C-13
D	Compatibility features.....	D-1
D.1	Postfix increment operator	D-1
D.2	<code>static</code> keyword.....	D-1
D.3	Access declarations	D-1
D.4	Implicit conversion from <code>const</code> strings.....	D-1
D.5	Standard C library headers	D-1
D.6	Old <code>iostreams</code> members.....	D-2
D.7	<code>char*</code> streams.....	D-3
D.7.1	Class <code>strstreambuf</code>	D-3
D.7.1.1	<code>strstreambuf</code> constructors.....	D-5
D.7.1.2	Member functions.....	D-6
D.7.1.3	<code>strstreambuf</code> overridden virtual functions.....	D-7
D.7.2	Class <code>istream</code>	D-10
D.7.2.1	<code>istream</code> constructors	D-10
D.7.2.2	Member functions.....	D-10
D.7.3	Class <code>ostream</code>	D-10
D.7.3.1	<code>ostream</code> constructors	D-11
D.7.3.2	Member functions.....	D-11
D.7.4	Class <code>strstream</code>	D-11
D.7.4.1	<code>strstream</code> constructors.....	D-12
D.7.4.2	<code>strstream</code> destructor	D-12
D.7.4.3	<code>strstream</code> operations.....	D-13

E Universal-character-names for identifiersE- 1