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

1.5 Syntax notation ..... 1−3

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 Trigraph sequences ..... 2−2

2.3 Preprocessing tokens ..... 2−2

2.4 Alternative tokens ..... 2−3

2.5 Tokens ..... 2−3

2.6 Comments ..... 2−4

2.7 Preprocessing numbers ..... 2−4

2.8 Identifiers ..... 2−4

2.9 Keywords ..... 2−5

2.10 Literals ..... 2−5
2.10.1 Integer literals ..... 2−6
2.10.2 Character literals ..... 2−7

2.10.3 Floating literals ....2−8
2.10.4 String literals ....2−9
2.10.5 Boolean literals....2−9

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....3−5
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 ....3−8
3.4.1 Unqualified name look up....3−8
3.4.2 Qualified name look up....3−11
3.4.2.1 Class members ....3−12
3.4.2.2 Namespace members....3−12
3.4.3 Elaborated type specifiers ....3−16
3.4.4 Class member access....3−17
3.4.5 Using directives and namespace aliases....3−17

3.5 Program and linkage ....3−17

3.6 Start and termination....3−19
3.6.1 Main function....3−19
3.6.2 Initialization of non-local objects....3−19
3.6.3 Termination....3−20

3.7 Storage duration ....3−21
3.7.1 Static storage duration....3−21
3.7.2 Automatic storage duration....3−22
3.7.3 Dynamic storage duration ....3−22
3.7.3.1 Allocation functions....3−22
3.7.3.2 Deallocation functions ....3−23
3.7.4 Duration of sub-objects....3−23

3.8 Object Lifetime ....3−23

3.9 Types....3−26
3.9.1 Fundamental types ....3−28
3.9.2 Compound types ....3−29
3.9.3 CV-qualifiers....3−30

3.10 Lvalues and rvalues....3−30

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 ....4−3
4.7 Integral conversions ....4−3
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 ....4−4
4.12 Base class conversion ....4−5
4.13 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 ....5−4
5.2.3 Explicit type conversion (functional notation) ....5−6
5.2.4 Class member access ....5−6
5.2.5 Increment and decrement ....5−7
5.2.6 Dynamic cast ....5−7
5.2.7 Type identification ....5−9
5.2.8 Static cast ....5−9
5.2.9 Reinterpret cast ....5−10
5.2.10 Const cast ....5−12
5.3 Unary expressions ....5−13
5.3.1 Unary operators ....5−13
5.3.2 Increment and decrement ....5−14
5.3.3 Sizeof ....5−14
5.3.4 New ....5−15
5.3.5 Delete ....5−17
5.4 Explicit type conversion (cast notation) ....5−18
5.5 Pointer-to-member operators ....5−19
5.6 Multiplicative operators ....5−19

5.7 Additive operators 5–19

5.8 Shift operators 5–21

5.9 Relational operators 5–21

5.10 Equality operators 5–22

5.11 Bitwise AND operator 5–22

5.12 Bitwise exclusive OR operator 5–23

5.13 Bitwise inclusive OR operator 5–23

5.14 Logical AND operator 5–23

5.15 Logical OR operator 5–23

5.16 Conditional operator 5–23

5.17 Assignment operators 5–24

5.18 Comma operator 5–25

5.19 Constant expressions 5–25

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 `if` statement 6–3
6.4.2 The `switch` statement 6–3

6.5 Iteration statements 6–3
6.5.1 The `while` statement 6–4
6.5.2 The `do` statement 6–4
6.5.3 The `for` statement 6–4

6.6 Jump statements 6–5
6.6.1 The `break` statement 6–5
6.6.2 The `continue` statement 6–5
6.6.3 The `return` statement 6–5
6.6.4 The `goto` statement 6–6

6.7 Declaration statement 6–6

6.8 Ambiguity resolution 6–6

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 `typedef` specifier 7–5
7.1.4 The `friend` specifier 7–6
7.1.5 Type specifiers 7–6
7.1.5.1 The *cv-qualifiers* 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 `using` declaration 7–15
7.3.4 Using directive 7–20

7.4 The `asm` declaration 7–22

7.5 Linkage specifications 7–22

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 8–7
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 8–17
8.5.2 Character arrays 8–20
8.5.3 References 8–20

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 `this` 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 9–11

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 10–9

11 Member access control 11–1

11.1 Access specifiers 11–2

11.2 Access specifiers for base classes 11–3

11.3 Access declarations 11–4

11.4 Friends 11–5

11.5 Protected member access 11–7

11.6 Access to virtual functions 11–8

11.7 Multiple access 11–8

12 Special member functions 12–1

12.1 Constructors 12–1

12.2 Temporary objects 12–2

12.3 Conversions 12–4
12.3.1 Conversion by constructor 12–4
12.3.2 Conversion functions 12–5

12.4 Destructors 12–7

12.5 Free store 12–9

12.6 Initialization ..... 12−11
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 ..... 13−5
13.3.1.1.1 Call to named function ..... 13−6
13.3.1.1.2 Call to object of class type ..... 13−7
13.3.1.2 Operators in expressions ..... 13−7
13.3.1.3 Initialization by user-defined conversions ..... 13−10
13.3.1.4 Initialization by constructor ..... 13−11
13.3.2 Viable functions ..... 13−11
13.3.3 Best Viable Function ..... 13−11
13.3.3.1 Implicit conversion sequences ..... 13−12
13.3.3.1.1 Standard conversion sequences ..... 13−13
13.3.3.1.2 User-defined conversion sequences ..... 13−14
13.3.3.1.3 Ellipsis conversion sequences ..... 13−14
13.3.3.1.4 Reference binding ..... 13−15
13.3.3.2 Ranking implicit conversion sequences ..... 13−15

13.4 Address of overloaded function ..... 13−17

13.5 Overloaded operators ..... 13−18
13.5.1 Unary operators ..... 13−19
13.5.2 Binary operators ..... 13−19
13.5.3 Assignment ..... 13−19
13.5.4 Function call ..... 13−20
13.5.5 Subscripting ..... 13−20
13.5.6 Class member access ..... 13−20
13.5.7 Increment and decrement ..... 13−21

13.6 Built-in operators ..... 13−21

14 Templates ..... 14−1

14.1 Template names ..... 14−2

14.2 Name resolution ..... 14−3
14.2.1 Locally declared names ..... 14−5
14.2.2 Names from the template’s enclosing scope ..... 14−6
14.2.3 Dependent names ..... 14−7
14.2.4 Non-local names declared within a template ..... 14−9

14.3 Template instantiation 14–9
14.3.1 Template linkage 14–10
14.3.2 Point of instantiation 14–10

14.4 Explicit instantiation 14–15

14.5 Template specialization 14–15

14.6 Class template specializations 14–17
14.6.1 Matching of class template specializations 14–18
14.6.2 Partial ordering of class template specializations 14–18
14.6.3 Member functions of class template specializations 14–19

14.7 Template parameters 14–20

14.8 Template arguments 14–21

14.9 Type equivalence 14–24

14.10 Function templates 14–24
14.10.1 Explicit template argument specification 14–24
14.10.2 Template argument deduction 14–25
14.10.3 Overload resolution 14–29
14.10.4 Overloading and linkage 14–31
14.10.5 Overloading and specialization 14–31
14.10.6 Partial ordering of function templates 14–32

14.11 Member function templates 14–33

14.12 Member class templates 14–34

14.13 Friends 14–34

14.14 Static members and variables 14–35

15 Exception handling 15–1

15.1 Throwing an exception 15–2

15.2 Constructors and destructors 15–3

15.3 Handling an exception 15–3

15.4 Exception specifications 15–5

15.5 Special functions 15–7
15.5.1 The `terminate()` function 15–7
15.5.2 The `unexpected()` function 15–8
15.5.3 The `uncaught_exception()` function 15–8

15.6 Exceptions and access 15–8

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 # operator 16–5
16.3.3 The ## 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 16–8

16.7 Null directive 16–9

16.8 Predefined macro names 16–9

17 Library introduction 17–1

17.1 Definitions 17–1

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 17–3
17.2.1.4 C Library 17–4
17.2.2 Other conventions 17–4
17.2.2.1 Type descriptions 17–4
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 17–6
17.2.2.1.3.3 Wide-character sequences 17–6
17.2.2.2 Functions within classes 17–7
17.2.2.3 Private members 17–7

17.3 Library-wide requirements 17–7
17.3.1 Library contents and organization 17–7
17.3.1.1 Library contents 17–7
17.3.1.2 Headers 17–15
17.3.1.3 Freestanding implementations 17–15
17.3.2 Using the library 17–16
17.3.2.1 Headers 17–16
17.3.2.2 Linkage 17–16
17.3.3 Constraints on programs 17–17
17.3.3.1 Reserved names 17–17
17.3.3.1.1 Macro names 17–17
17.3.3.1.2 Global names 17–17

17.3.3.1.3 External linkage ....17−17
17.3.3.2 Headers ....17−18
17.3.3.3 Derived classes ....17−18
17.3.3.4 Replacement functions ....17−18
17.3.3.5 Handler functions ....17−18
17.3.3.6 Other functions ....17−19
17.3.3.7 Function arguments ....17−19
17.3.3.8 Required paragraph ....17−19
17.3.4 Conforming implementations ....17−19
17.3.4.1 Headers ....17−20
17.3.4.2 Restrictions on macro definitions ....17−20
17.3.4.3 Global functions ....17−20
17.3.4.4 Member functions ....17−20
17.3.4.5 Reentrancy ....17−21
17.3.4.6 Protection within classes ....17−21
17.3.4.7 Derived classes ....17−21
17.3.4.8 Restrictions on exception handling ....17−21

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 `numeric_limits` ....18−2
18.2.1.2 `numeric_limits` members ....18−3
18.2.1.3 Type `float_round_style` ....18−8
18.2.1.4 `numeric_limits` 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 `bad_alloc` ....18−14
18.4.2.2 Type `new_handler` ....18−15
18.4.2.3 `set_new_handler` ....18−15

18.5 Type identification ....18−15
18.5.1 Class `type_info` ....18−15
18.5.2 Class `bad_cast` ....18−16
18.5.3 Class `bad_typeid` ....18−17

18.6 Exception handling ....18−17
18.6.1 Class `exception` ....18−18
18.6.2 Violating *`exception-specifications`* ....18−18
18.6.2.1 Class `bad_exception` ....18−18
18.6.2.2 Type `unexpected_handler` ....18−19
18.6.2.3 `set_unexpected` ....18−19
18.6.2.4 `unexpected` ....18−19

18.6.3 Abnormal termination ..... 18−20
18.6.3.1 Type `terminate_handler` ..... 18−20
18.6.3.2 `set_terminate` ..... 18−20
18.6.3.3 `terminate` ..... 18−20
18.6.4 `uncaught_exception` ..... 18−20

18.7 Other runtime support ..... 18−20

19 Diagnostics library ..... 19−1

19.1 Exception classes ..... 19−1
19.1.1 Class `logic_error` ..... 19−2
19.1.2 Class `domain_error` ..... 19−2
19.1.3 Class `invalid_argument` ..... 19−2
19.1.4 Class `length_error` ..... 19−2
19.1.5 Class `out_of_range` ..... 19−3
19.1.6 Class `runtime_error` ..... 19−3
19.1.7 Class `range_error` ..... 19−3
19.1.8 Class `overflow_error` ..... 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 Allocator requirements ..... 20−2

20.2 Utility components ..... 20−4
20.2.1 Operators ..... 20−5
20.2.2 Pairs ..... 20−5

20.3 Function objects ..... 20−6
20.3.1 Base ..... 20−8
20.3.2 Arithmetic operations ..... 20−8
20.3.3 Comparisons ..... 20−9
20.3.4 Logical operations ..... 20−10
20.3.5 Negators ..... 20−10
20.3.6 Binders ..... 20−11
20.3.6.1 Template class `binder1st` ..... 20−11
20.3.6.2 `bind1st` ..... 20−11
20.3.6.3 Template class `binder2nd` ..... 20−11
20.3.6.4 `bind2nd` ..... 20−12
20.3.7 Adaptors for pointers to functions ..... 20−12

20.4 Memory ..... 20−13
20.4.1 The default allocator ..... 20−13
20.4.1.1 `allocator` members ..... 20−14
20.4.1.2 `allocator` globals ..... 20−15
20.4.1.3 Example `allocator` ..... 20−15

20.4.2 Raw storage iterator 20–16
20.4.3 Temporary buffers 20–17
20.4.4 Specialized algorithms 20–18
20.4.4.1 uninitialized_copy 20–18
20.4.4.2 uninitialized_fill 20–18
20.4.4.3 uninitialized_fill_n 20–18
20.4.5 Template class auto_ptr 20–18
20.4.5.1 auto_ptr constructors 20–19
20.4.5.2 auto_ptr members 20–19
20.4.6 C Library 20–20

20.5 Date and time 20–20

21 Strings library 21–1

21.1 String classes 21–1
21.1.1 Template class basic_string 21–3
21.1.1.1 Template class string_char_traits 21–3
21.1.1.2 string_char_traits members 21–4
21.1.1.3 Template class basic_string 21–5
21.1.1.4 basic_string constructors 21–8
21.1.1.5 basic_string iterator support 21–11
21.1.1.6 basic_string capacity 21–11
21.1.1.7 basic_string element access 21–12
21.1.1.8 basic_string modifiers 21–12
21.1.1.8.1 basic_string::operator+= 21–12
21.1.1.8.2 basic_string::append 21–13
21.1.1.8.3 basic_string::assign 21–13
21.1.1.8.4 basic_string::insert 21–14
21.1.1.8.5 basic_string::erase 21–15
21.1.1.8.6 basic_string::replace 21–16
21.1.1.8.7 basic_string::copy 21–17
21.1.1.8.8 basic_string::swap 21–18
21.1.1.9 basic_string string operations 21–18
21.1.1.9.1 basic_string::find 21–18
21.1.1.9.2 basic_string::rfind 21–19
21.1.1.9.3 basic_string::find_first_of 21–19
21.1.1.9.4 basic_string::find_last_of 21–20
21.1.1.9.5 basic_string::find_first_not_of 21–20
21.1.1.9.6 basic_string::find_last_not_of 21–21
21.1.1.9.7 basic_string::substr 21–21
21.1.1.9.8 basic_string::compare 21–22
21.1.1.10 basic_string non-member functions 21–23
21.1.1.10.1 operator+ 21–23
21.1.1.10.2 operator== 21–23
21.1.1.10.3 operator!= 21–24
21.1.1.10.4 operator< 21–24
21.1.1.10.5 operator> 21–25
21.1.1.10.6 operator<= 21–25
21.1.1.10.7 operator>= 21–25
21.1.1.10.8 swap 21–26
21.1.1.10.9 Inserters and extractors 21–26
21.1.2 Class string 21–27
21.1.3 string_char_traits<char> members 21–28

21.1.4 Class `wstring` ..... 21−28
21.1.5 `string_char_traits<wchar_t>` members ..... 21−29

21.2 Null-terminated sequence utilities ..... 21−30

22 Localization library ..... 22−1

22.1 Locales ..... 22−1
22.1.1 Class `locale` ..... 22−3
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 ..... 22−10
22.1.3.1 Character classification ..... 22−10
22.1.3.2 Character conversions ..... 22−11

22.2 Standard `locale` categories ..... 22−11
22.2.1 The `ctype` category ..... 22−11
22.2.1.1 Template class `ctype` ..... 22−12
22.2.1.1.1 `ctype` members ..... 22−13
22.2.1.1.2 `ctype` virtual functions ..... 22−14
22.2.1.2 Template class `ctype_byname` ..... 22−15
22.2.1.3 `ctype` specializations ..... 22−15
22.2.1.3.1 `ctype<char>` destructor ..... 22−16
22.2.1.3.2 `ctype<char>` members ..... 22−16
22.2.1.3.3 `ctype<char>` static members ..... 22−18
22.2.1.3.4 `ctype<char>` virtual functions ..... 22−18
22.2.1.4 Class ctype_byname<char> ..... 22−18
22.2.1.5 Template class `codecvt` ..... 22−18
22.2.1.5.1 `codecvt` members ..... 22−19
22.2.1.5.2 `codecvt` virtual functions ..... 22−20
22.2.1.6 Template class `codecvt_byname` ..... 22−21
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` ..... 22−23
22.2.2.2.1 `num_put` members ..... 22−24
22.2.2.2.2 `num_put` virtual functions ..... 22−24
22.2.3 The numeric punctuation facet ..... 22−25
22.2.3.1 Template class `numpunct` ..... 22−25
22.2.3.1.1 `numpunct` members ..... 22−26
22.2.3.1.2 `numpunct` virtual functions ..... 22−26
22.2.3.2 Template class `numpunct_byname` ..... 22−27
22.2.4 The collate category ..... 22−27
22.2.4.1 Template class `collate` ..... 22−27
22.2.4.1.1 `collate` members ..... 22−28
22.2.4.1.2 `collate` virtual functions ..... 22−28

22.2.4.2 Template class `collate_byname` 22–29
22.2.5 The time category 22–29
22.2.5.1 Template class `time_get` 22–29
22.2.5.1.1 `time_get` members 22–30
22.2.5.1.2 `time_get` virtual functions 22–31
22.2.5.2 Template class `time_get_byname` 22–32
22.2.5.3 Template class `time_put` 22–32
22.2.5.3.1 `time_put` members 22–32
22.2.5.3.2 `time_put` virtual functions 22–33
22.2.5.4 Template class `time_put_byname` 22–33
22.2.6 The monetary category 22–33
22.2.6.1 Template class `money_get` 22–33
22.2.6.1.1 `money_get` members 22–34
22.2.6.1.2 `money_get` virtual functions 22–34
22.2.6.2 Template class `money_put` 22–35
22.2.6.2.1 `money_put` members 22–35
22.2.6.2.2 `money_put` virtual functions 22–35
22.2.6.3 Template class `moneypunct` 22–36
22.2.6.3.1 `moneypunct` members 22–37
22.2.6.3.2 `moneypunct` virtual functions 22–37
22.2.6.4 Template class `moneypunct_byname` 22–38
22.2.7 The message retrieval category 22–38
22.2.7.1 Template class `messages` 22–38
22.2.7.1.1 `messages` members 22–39
22.2.7.1.2 `messages` virtual functions 22–39
22.2.7.2 Template class `messages_byname` 22–40
22.2.8 Program-defined facets 22–40

22.3 C Library Locales 22–43

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 `bitset` 23–10
23.2.1.1 `bitset` constructors 23–12
23.2.1.2 `bitset` members 23–12
23.2.1.3 `bitset` operators 23–15
23.2.2 Template class `deque` 23–16
23.2.2.1 `deque` types 23–17
23.2.2.2 `deque` constructors, copy, and assignment 23–17
23.2.2.3 `deque` iterator support 23–18
23.2.2.4 `deque` capacity 23–18
23.2.2.5 `deque` element access 23–18
23.2.2.6 `deque` modifiers 23–18
23.2.2.7 `deque` specialized algorithms 23–18
23.2.3 Template class `list` 23–19
23.2.3.1 `list` types 23–21
23.2.3.2 `list` constructors, copy, and assignment 23–21
23.2.3.3 `list` iterator support 23–21
23.2.3.4 `list` capacity 23–21

23.2.3.5 `list` element access ....23−21
23.2.3.6 `list` modifiers ....23−21
23.2.3.7 `list` operations ....23−22
23.2.3.8 `list` specialized algorithms ....23−23
23.2.4 Container adapters ....23−23
23.2.4.1 Template class `queue` ....23−23
23.2.4.2 Template class `priority_queue` ....23−24
23.2.4.2.1 `priority_queue` constructors ....23−24
23.2.4.2.2 `priority_queue` members ....23−25
23.2.4.3 Template class `stack` ....23−25
23.2.5 Template class `vector` ....23−26
23.2.5.1 `vector` types ....23−27
23.2.5.2 `vector` constructors, copy, and assignment ....23−27
23.2.5.3 `vector` iterator support ....23−28
23.2.5.4 `vector` capacity ....23−28
23.2.5.5 `vector` element access ....23−28
23.2.5.6 `vector` modifiers ....23−28
23.2.5.7 `vector` specialized algorithms ....23−29
23.2.6 Class `vector<bool>` ....23−29

23.3 Associative containers ....23−31
23.3.1 Template class `map` ....23−32
23.3.1.1 `map` types ....23−34
23.3.1.2 `map` constructors, copy, and assignment ....23−34
23.3.1.3 `map` iterator support ....23−34
23.3.1.4 `map` capacity ....23−35
23.3.1.5 `map` element access ....23−35
23.3.1.6 `map` modifiers ....23−35
23.3.1.7 `map` observers ....23−35
23.3.1.8 `map` operations ....23−35
23.3.1.9 `map` specialized algorithms ....23−35
23.3.2 Template class `multimap` ....23−35
23.3.2.1 `multimap` specialized algorithms ....23−37
23.3.3 Template class `set` ....23−37
23.3.3.1 `set` types ....23−39
23.3.3.2 `set` constructors, copy, and assignment ....23−39
23.3.3.3 `set` iterator support ....23−39
23.3.3.4 `set` capacity ....23−39
23.3.3.5 `set` modifiers ....23−39
23.3.3.6 `set` observers ....23−39
23.3.3.7 `set` operations ....23−39
23.3.3.8 `set` specialized algorithms ....23−39
23.3.4 Template class `multiset` ....23−39
23.3.4.1 `multiset` specialized algorithms ....23−41

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 ....24−4
24.1.5 Random access iterators ....24−5
24.1.6 Iterator tags ....24−6

24.2 Iterator primitives 24–11
24.2.1 Standard iterator tags 24–11
24.2.2 Basic iterators 24–11
24.2.3 `iterator_category` 24–12
24.2.4 `value_type` 24–13
24.2.5 `distance_type` 24–13
24.2.6 Iterator operations 24–13

24.3 Predefined iterators 24–14
24.3.1 Reverse iterators 24–14
24.3.1.1 Template class `reverse_bidirectional_iterator` 24–14
24.3.1.2 `reverse_bidirectional_iterator` operations 24–15
24.3.1.2.1 `reverse_bidirectional_iterator` constructor 24–15
24.3.1.2.2 Conversion 24–15
24.3.1.2.3 `operator*` 24–15
24.3.1.2.4 `operator->` 24–15
24.3.1.2.5 `operator++` 24–16
24.3.1.2.6 `operator--` 24–16
24.3.1.2.7 `operator==` 24–16
24.3.1.3 Template class `reverse_iterator` 24–16
24.3.1.4 `reverse_iterator` operations 24–17
24.3.1.4.1 `reverse_iterator` constructor 24–17
24.3.1.4.2 Conversion 24–18
24.3.1.4.3 `operator*` 24–18
24.3.1.4.4 `operator->` 24–18
24.3.1.4.5 `operator++` 24–18
24.3.1.4.6 `operator--` 24–18
24.3.1.4.7 `operator+` 24–19
24.3.1.4.8 `operator+=` 24–19
24.3.1.4.9 `operator-` 24–19
24.3.1.4.10 `operator-=` 24–19
24.3.1.4.11 `operator[]` 24–19
24.3.1.4.12 `operator==` 24–19
24.3.1.4.13 `operator<` 24–20
24.3.1.4.14 `operator-` 24–20
24.3.1.4.15 `operator==` 24–20
24.3.2 Insert iterators 24–20
24.3.2.1 Template class `back_insert_iterator` 24–21
24.3.2.2 `back_insert_iterator` operations 24–21
24.3.2.2.1 `back_insert_iterator` constructor 24–21
24.3.2.2.2 `back_insert_iterator::operator=` 24–21
24.3.2.2.3 `back_insert_iterator::operator*` 24–21
24.3.2.2.4 `back_insert_iterator::operator++` 24–21
24.3.2.2.5 `back_inserter` 24–22
24.3.2.3 Template class `front_insert_iterator` 24–22
24.3.2.4 `front_insert_iterator` operations 24–22
24.3.2.4.1 `front_insert_iterator` constructor 24–22
24.3.2.4.2 `front_insert_iterator::operator=` 24–22
24.3.2.4.3 `front_insert_iterator::operator*` 24–22
24.3.2.4.4 `front_insert_iterator::operator++` 24–23
24.3.2.4.5 `front_inserter` 24–23
24.3.2.5 Template class `insert_iterator` 24–23
24.3.2.6 `insert_iterator` operations 24–23
24.3.2.6.1 `insert_iterator` constructor 24–23

24.3.2.6.2 `insert_iterator::operator=` 24–23
24.3.2.6.3 `insert_iterator::operator*` 24–24
24.3.2.6.4 `insert_iterator::operator++` 24–24
24.3.2.6.5 `inserter` 24–24

24.4 Stream iterators 24–24
24.4.1 Template class `istream_iterator` 24–24
24.4.2 Template class `ostream_iterator` 24–25
24.4.3 Template class `istreambuf_iterator` 24–25
24.4.3.1 Template class `istreambuf_iterator::proxy` 24–27
24.4.3.2 `istreambuf_iterator` constructors 24–27
24.4.3.3 `istreambuf_iterator::operator*` 24–28
24.4.3.4 `istreambuf_iterator::operator++` 24–28
24.4.3.5 `istreambuf_iterator::equal` 24–28
24.4.3.6 `iterator_category` 24–28
24.4.3.7 `operator==` 24–28
24.4.3.8 `operator!=` 24–28
24.4.4 Template class `ostreambuf_iterator` 24–29
24.4.4.1 `ostreambuf_iterator` constructors 24–29
24.4.4.2 `ostreambuf_iterator` operations 24–29
24.4.4.3 `ostreambuf_iterator` non-member operations 24–30

25 Algorithms library 25–1

25.1 Non-modifying sequence operations 25–9
25.1.1 For each 25–9
25.1.2 Find 25–9
25.1.3 Find End 25–10
25.1.4 Find First 25–10
25.1.5 Adjacent find 25–10
25.1.6 Count 25–11
25.1.7 Mismatch 25–11
25.1.8 Equal 25–12
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 25–16
25.2.8 Unique 25–16
25.2.9 Reverse 25–17
25.2.10 Rotate 25–17
25.2.11 Random shuffle 25–18
25.2.12 Partitions 25–18

25.3 Sorting and related operations 25–19
25.3.1 Sorting 25–19
25.3.1.1 `sort` 25–19
25.3.1.2 `stable_sort` 25–20
25.3.1.3 `partial_sort` 25–20

25.3.1.4 `partial_sort_copy` 25–20
25.3.2 Nth element 25–21
25.3.3 Binary search 25–21
25.3.3.1 `lower_bound` 25–21
25.3.3.2 `upper_bound` 25–22
25.3.3.3 `equal_range` 25–22
25.3.3.4 `binary_search` 25–23
25.3.4 Merge 25–23
25.3.5 Set operations on sorted structures 25–24
25.3.5.1 `includes` 25–24
25.3.5.2 `set_union` 25–24
25.3.5.3 `set_intersection` 25–25
25.3.5.4 `set_difference` 25–25
25.3.5.5 `set_symmetric_difference` 25–26
25.3.6 Heap operations 25–26
25.3.6.1 `push_heap` 25–27
25.3.6.2 `pop_heap` 25–27
25.3.6.3 `make_heap` 25–27
25.3.6.4 `sort_heap` 25–27
25.3.7 Minimum and maximum 25–28
25.3.8 Lexicographical comparison 25–28
25.3.9 Permutation generators 25–29

25.4 C library algorithms 25–30

26 Numerics library 26–1

26.1 Numeric type requirements 26–1

26.2 Complex numbers 26–2
26.2.1 Template class `complex` 26–3
26.2.2 `complex` specializations 26–4
26.2.3 `complex` member functions 26–5
26.2.4 `complex` member operators 26–5
26.2.5 `complex` non-member operations 26–6
26.2.6 `complex` value operations 26–7
26.2.7 `complex` transcendentals 26–8

26.3 Numeric arrays 26–8
26.3.1 Template class `valarray` 26–11
26.3.1.1 `valarray` constructors 26–12
26.3.1.2 `valarray` assignment 26–13
26.3.1.3 `valarray` element access 26–14
26.3.1.4 `valarray` subset operations 26–14
26.3.1.5 `valarray` unary operators 26–14
26.3.1.6 `valarray` computed assignment 26–15
26.3.1.7 `valarray` member functions 26–16
26.3.2 `valarray` non-member operations 26–17
26.3.2.1 `valarray` binary operators 26–17
26.3.2.2 `valarray` comparison operators 26–18
26.3.2.3 `valarray` min and max functions 26–19
26.3.2.4 `valarray` transcendentals 26–19
26.3.3 Class `slice` 26–20
26.3.3.1 `slice` constructors 26–20

26.3.3.2 `slice` access functions 26–21
26.3.4 Template class `slice_array` 26–21
26.3.4.1 `slice_array` constructors 26–22
26.3.4.2 `slice_array` assignment 26–22
26.3.4.3 `slice_array` computed assignment 26–22
26.3.4.4 `slice_array` fill function 26–22
26.3.5 The `gslice` class 26–22
26.3.5.1 `gslice constructors` 26–24
26.3.5.2 `gslice` access functions 26–24
26.3.6 Template class `gslice_array` 26–24
26.3.6.1 `gslice_array` constructors 26–25
26.3.6.2 `gslice_array` assignment 26–25
26.3.6.3 `gslice_array computed assignment` 26–25
26.3.6.4 `gslice_array` fill function 26–25
26.3.7 Template class `mask_array` 26–26
26.3.7.1 `mask_array` constructors 26–26
26.3.7.2 `mask_array` assignment 26–26
26.3.7.3 `mask_array` computed assignment 26–27
26.3.7.4 `mask_array` fill function 26–27
26.3.8 Template class `indirect_array` 26–27
26.3.8.1 `indirect_array` constructors 26–28
26.3.8.2 `indirect_array` assignment 26–28
26.3.8.3 `indirect_array` computed assignment 26–28
26.3.8.4 `indirect_array` fill function 26–29

26.4 Generalized numeric operations 26–29
26.4.1 Accumulate 26–29
26.4.2 Inner product 26–30
26.4.3 Partial sum 26–30
26.4.4 Adjacent difference 26–30

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 Type requirements 27–2
27.1.2.1 Type *CHAR_T* 27–2
27.1.2.2 Type *INT_T* 27–2
27.1.2.3 Type *OFF_T* 27–2
27.1.2.4 Type *POS_T* 27–3
27.1.2.5 Type *SZ_T* 27–3
27.1.2.6 Type *STATE_T* 27–3

27.2 Forward declarations 27–4

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 Template struct `ios_traits` 27–8

27.4.2.1 ios_traits value functions ..... 27−9
27.4.2.2 ios_traits test functions ..... 27−10
27.4.2.3 ios_traits conversion functions ..... 27−10
27.4.3 Class ios_base ..... 27−11
27.4.3.1 Types ..... 27−13
27.4.3.1.1 Class ios_base::failure ..... 27−13
27.4.3.1.2 Type ios_base::fmtflags ..... 27−14
27.4.3.1.3 Type ios_base::iostate ..... 27−14
27.4.3.1.4 Type ios_base::openmode ..... 27−15
27.4.3.1.5 Type ios_base::seekdir ..... 27−15
27.4.3.1.6 Class ios_base::Init ..... 27−15
27.4.3.2 ios_base fmtflags state functions ..... 27−16
27.4.3.3 ios_base locale functions ..... 27−17
27.4.3.4 ios_base storage functions ..... 27−17
27.4.3.5 ios_base constructors ..... 27−18
27.4.4 Template class basic_ios ..... 27−18
27.4.4.1 basic_ios constructors ..... 27−19
27.4.4.2 Member functions ..... 27−20
27.4.4.3 basic_ios iostate flags functions ..... 27−22
27.4.5 ios_base manipulators ..... 27−23
27.4.5.1 fmtflags manipulators ..... 27−23
27.4.5.2 adjustfield manipulators ..... 27−24
27.4.5.3 basefield manipulators ..... 27−24
27.4.5.4 floatfield manipulators ..... 27−25

27.5 Stream buffers ..... 27−25
27.5.1 Stream buffer requirements ..... 27−25
27.5.2 Template class basic_streambuf<charT,traits> ..... 27−26
27.5.2.1 basic_streambuf constructors ..... 27−28
27.5.2.2 basic_streambuf public member functions ..... 27−28
27.5.2.2.1 Locales ..... 27−28
27.5.2.2.2 Buffer management and positioning ..... 27−29
27.5.2.2.3 Get area ..... 27−29
27.5.2.2.4 Putback ..... 27−30
27.5.2.2.5 Put area ..... 27−30
27.5.2.3 basic_streambuf protected member functions ..... 27−30
27.5.2.3.1 Get area access ..... 27−30
27.5.2.3.2 Put area access ..... 27−31
27.5.2.4 basic_streambuf virtual functions ..... 27−31
27.5.2.4.1 Locales ..... 27−31
27.5.2.4.2 Buffer management and positioning ..... 27−31
27.5.2.4.3 Get area ..... 27−32
27.5.2.4.4 Putback ..... 27−34
27.5.2.4.5 Put area ..... 27−34

27.6 Formatting and manipulators ..... 27−35
27.6.1 Input streams ..... 27−36
27.6.1.1 Template class basic_istream ..... 27−36
27.6.1.1.1 basic_istream constructors ..... 27−38
27.6.1.1.2 basic_istream prefix and suffix ..... 27−38
27.6.1.2 Formatted input functions ..... 27−39
27.6.1.2.1 Common requirements ..... 27−39
27.6.1.2.2 basic_istream::operator>> ..... 27−41
27.6.1.3 Unformatted input functions ..... 27−44

27.6.1.4 Standard `basic_istream` manipulators ....27−47
27.6.2 Output streams ....27−47
27.6.2.1 Template class `basic_ostream` ....27−47
27.6.2.2 `basic_ostream` constructors ....27−49
27.6.2.3 `basic_ostream` prefix and suffix functions ....27−49
27.6.2.4 Formatted output functions ....27−50
27.6.2.4.1 Common requirements ....27−50
27.6.2.4.2 `basic_ostream::operator<<` ....27−53
27.6.2.5 Unformatted output functions ....27−55
27.6.2.6 Standard `basic_ostream` manipulators ....27−55
27.6.3 Standard manipulators ....27−56

27.7 String-based streams ....27−57
27.7.1 Template class `basic_stringbuf` ....27−58
27.7.1.1 `basic_stringbuf` constructors ....27−58
27.7.1.2 Member functions ....27−59
27.7.1.3 Overridden virtual functions ....27−60
27.7.2 Template class `basic_istringstream` ....27−62
27.7.2.1 `basic_istringstream` constructors ....27−63
27.7.2.2 Member functions ....27−63
27.7.2.3 Class `basic_ostringstream` ....27−63
27.7.2.4 `basic_ostringstream` constructors ....27−64
27.7.2.5 Member functions ....27−64

27.8 File-based streams ....27−64
27.8.1 File streams ....27−64
27.8.1.1 Template class `basic_filebuf` ....27−65
27.8.1.2 `basic_filebuf` constructors ....27−66
27.8.1.3 Member functions ....27−67
27.8.1.4 Overridden virtual functions ....27−68
27.8.1.5 Template class `basic_ifstream` ....27−70
27.8.1.6 `basic_ifstream` constructors ....27−71
27.8.1.7 Member functions ....27−71
27.8.1.8 Template class `basic_ofstream` ....27−72
27.8.1.9 `basic_ofstream` constructors ....27−72
27.8.1.10 Member functions ....27−72
27.8.2 C Library files ....27−73

A Grammar summary ....A−1

A.1 Keywords ....A−1

A.2 Lexical conventions ....A−1

A.3 Basic concepts ....A−4

A.4 Expressions ....A−4

A.5 Statements ....A−8

A.6 Declarations ....A−8

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−16

B Implementation quantities ..... B−1

C Compatibility ..... C−1

C.1 Extensions ..... C−1
C.1.1 C++ features available in 1985 ..... C−1
C.1.2 C++ features added since 1985 ..... C−2

C.2 C++ and ISO C ..... C−2
C.2.1 Clause 2: lexical conventions ..... C−2
C.2.2 Clause 3: basic concepts ..... C−3
C.2.3 Clause 5: expressions ..... C−5
C.2.4 Clause 6: statements ..... C−5
C.2.5 Clause 7: declarations ..... C−6
C.2.6 Clause 8: declarators ..... C−8
C.2.7 Clause 9: classes ..... C−9
C.2.8 Clause 12: special member functions ..... C−10
C.2.9 Clause 16: preprocessing directives ..... C−11

C.3 Anachronisms ..... C−11
C.3.1 Old style function definitions ..... C−11
C.3.2 Old style base class initializer ..... C−12
C.3.3 Assignment to `this` ..... C−12
C.3.4 Cast of bound pointer ..... C−12
C.3.5 Nonnested classes ..... C−12

C.4 Standard C library ..... C−13
C.4.1 Modifications to headers ..... C−15
C.4.2 Modifications to definitions ..... C−15
C.4.2.1 Type `wchar_t` ..... C−15
C.4.2.2 Header `<iso646.h>` ..... C−15
C.4.2.3 Macro `NULL` ..... C−15
C.4.3 Modifications to declarations ..... C−15
C.4.4 Modifications to behavior ..... C−15
C.4.4.1 Macro `offsetof(`*`type, member-designator`*`)` ..... C−15
C.4.4.2 Memory allocation functions ..... C−16

D Compatibility features ..... D−1

D.1 Postfix increment operator ..... D−1

D.2 static keyword ..... D−1

D.3 Access declarations ..... D−1

D.4 Standard C library headers ..... D−1

D.5 Old iostreams members ..... D−2

D.6 `char*` streams ..... D−3
D.6.1 Class `strstreambuf` ..... D−3
D.6.1.1 `strstreambuf` constructors ..... D−5
D.6.1.2 Member functions ..... D−6
D.6.1.3 `strstreambuf` overridden virtual functions ..... D−7
D.6.2 Template class `istrstream` ..... D−10
D.6.2.1 `istrstream` constructors ..... D−10
D.6.2.2 Member functions ..... D−10
D.6.3 Template class `ostrstream` ..... D−11
D.6.3.1 `ostrstream` constructors ..... D−11
D.6.3.2 Member functions ..... D−11