

Contents

Appendices E through H are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).

Preface

xix

I	Introduction to Computers, the Internet and the Web	I
1.1	Introduction	2
1.2	Computers and the Internet in Industry and Research	2
1.3	Hardware and Software	5
1.3.1	Moore's Law	6
1.3.2	Computer Organization	6
1.4	Data Hierarchy	7
1.5	Programming Languages	9
1.6	The C Programming Language	10
1.7	C Standard Library	12
1.8	C++ and Other C-Based Languages	13
1.9	Object Technology	14
1.10	Typical C Program Development Environment	16
1.10.1	Phase 1: Creating a Program	16
1.10.2	Phases 2 and 3: Preprocessing and Compiling a C Program	16
1.10.3	Phase 4: Linking	18
1.10.4	Phase 5: Loading	18
1.10.5	Phase 6: Execution	18
1.10.6	Problems That May Occur at Execution Time	18
1.10.7	Standard Input, Standard Output and Standard Error Streams	18
1.11	Test-Driving a C Application in Windows, Linux and Mac OS X	19
1.11.1	Running a C Application from the Windows Command Prompt	20
1.11.2	Running a C Application Using GNU C with Linux	22
1.11.3	Running a C Application Using GNU C with Mac OS X	25
1.12	Operating Systems	27
1.12.1	Windows—A Proprietary Operating System	28
1.12.2	Linux—An Open-Source Operating System	28
1.12.3	Apple's Mac OS X; Apple's iOS for iPhone®, iPad® and iPod Touch® Devices	29
1.12.4	Google's Android	29

1.13	The Internet and World Wide Web	30
1.14	Some Key Software Development Terminology	31
1.15	Keeping Up-to-Date with Information Technologies	33
1.16	Web Resources	34
2	Introduction to C Programming	40
2.1	Introduction	41
2.2	A Simple C Program: Printing a Line of Text	41
2.3	Another Simple C Program: Adding Two Integers	45
2.4	Memory Concepts	49
2.5	Arithmetic in C	50
2.6	Decision Making: Equality and Relational Operators	54
2.7	Secure C Programming	58
3	Structured Program Development in C	70
3.1	Introduction	71
3.2	Algorithms	71
3.3	Pseudocode	71
3.4	Control Structures	72
3.5	The if Selection Statement	74
3.6	The if...else Selection Statement	75
3.7	The while Repetition Statement	79
3.8	Formulating Algorithms Case Study 1: Counter-Controlled Repetition	80
3.9	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 2: Sentinel-Controlled Repetition	82
3.10	Formulating Algorithms with Top-Down, Stepwise Refinement Case Study 3: Nested Control Statements	89
3.11	Assignment Operators	93
3.12	Increment and Decrement Operators	93
3.13	Secure C Programming	96
4	C Program Control	114
4.1	Introduction	115
4.2	Repetition Essentials	115
4.3	Counter-Controlled Repetition	116
4.4	for Repetition Statement	117
4.5	for Statement: Notes and Observations	120
4.6	Examples Using the for Statement	121
4.7	switch Multiple-Selection Statement	124
4.8	do...while Repetition Statement	130
4.9	break and continue Statements	132
4.10	Logical Operators	134
4.11	Confusing Equality (==) and Assignment (=) Operators	137

4.12	Structured Programming Summary	138
4.13	Secure C Programming	143
5	C Functions	158
5.1	Introduction	159
5.2	Program Modules in C	159
5.3	Math Library Functions	160
5.4	Functions	162
5.5	Function Definitions	162
5.6	Function Prototypes: A Deeper Look	166
5.7	Function Call Stack and Stack Frames	169
5.8	Headers	172
5.9	Passing Arguments By Value and By Reference	173
5.10	Random Number Generation	174
5.11	Example: A Game of Chance	179
5.12	Storage Classes	182
5.13	Scope Rules	184
5.14	Recursion	187
5.15	Example Using Recursion: Fibonacci Series	191
5.16	Recursion vs. Iteration	194
5.17	Secure C Programming	197
6	C Arrays	216
6.1	Introduction	217
6.2	Arrays	217
6.3	Defining Arrays	218
6.4	Array Examples	219
6.5	Passing Arrays to Functions	232
6.6	Sorting Arrays	236
6.7	Case Study: Computing Mean, Median and Mode Using Arrays	239
6.8	Searching Arrays	244
6.9	Multidimensional Arrays	249
6.10	Variable-Length Arrays	256
6.11	Secure C Programming	259
7	C Pointers	277
7.1	Introduction	278
7.2	Pointer Variable Definitions and Initialization	278
7.3	Pointer Operators	279
7.4	Passing Arguments to Functions by Reference	282
7.5	Using the const Qualifier with Pointers	284
7.5.1	Converting a String to Uppercase Using a Non-Constant Pointer to Non-Constant Data	287

7.5.2	Printing a String One Character at a Time Using a Non-Constant Pointer to Constant Data	288
7.5.3	Attempting to Modify a Constant Pointer to Non-Constant Data	290
7.5.4	Attempting to Modify a Constant Pointer to Constant Data	291
7.6	Bubble Sort Using Pass-by-Reference	291
7.7	sizeof Operator	294
7.8	Pointer Expressions and Pointer Arithmetic	297
7.9	Relationship between Pointers and Arrays	299
7.10	Arrays of Pointers	303
7.11	Case Study: Card Shuffling and Dealing Simulation	304
7.12	Pointers to Functions	309
7.13	Secure C Programming	314
8	C Characters and Strings	334
8.1	Introduction	335
8.2	Fundamentals of Strings and Characters	335
8.3	Character-Handling Library	337
8.3.1	Functions isdigit, isalpha, isalnum and isxdigit	338
8.3.2	Functions islower, isupper, tolower and toupper	340
8.3.3	Functions isspace, iscntrl, ispunct, isprint and isgraph	341
8.4	String-Conversion Functions	342
8.4.1	Function strtod	343
8.4.2	Function strtol	344
8.4.3	Function strtoul	345
8.5	Standard Input/Output Library Functions	346
8.5.1	Functions fgets and putchar	346
8.5.2	Function getchar	348
8.5.3	Function sprintf	349
8.5.4	Function sscanf	349
8.6	String-Manipulation Functions of the String-Handling Library	350
8.6.1	Functions strcpy and strncpy	351
8.6.2	Functions strcat and strncat	352
8.7	Comparison Functions of the String-Handling Library	353
8.8	Search Functions of the String-Handling Library	354
8.8.1	Function strchr	355
8.8.2	Function strcspn	356
8.8.3	Function strpbrk	357
8.8.4	Function strrchr	357
8.8.5	Function strspn	358
8.8.6	Function strstr	358
8.8.7	Function strtok	359
8.9	Memory Functions of the String-Handling Library	360
8.9.1	Function memcpy	361
8.9.2	Function memmove	362
8.9.3	Function memcmp	363

8.9.4	Function memchr	363
8.9.5	Function memset	364
8.10	Other Functions of the String-Handling Library	365
8.10.1	Function strerror	365
8.10.2	Function strlen	365
8.11	Secure C Programming	366

9 C Formatted Input/Output

9.1	Introduction	379
9.2	Streams	380
9.3	Formatting Output with printf	380
9.4	Printing Integers	381
9.5	Printing Floating-Point Numbers	382
9.6	Printing Strings and Characters	384
9.7	Other Conversion Specifiers	385
9.8	Printing with Field Widths and Precision	386
9.9	Using Flags in the printf Format Control String	388
9.10	Printing Literals and Escape Sequences	391
9.11	Reading Formatted Input with scanf	391
9.12	Secure C Programming	398

10 C Structures, Unions, Bit Manipulation and Enumerations

10.1	Introduction	405
10.2	Structure Definitions	406
10.2.1	Self-Referential Structures	407
10.2.2	Defining Variables of Structure Types	407
10.2.3	Structure Tag Names	408
10.2.4	Operations That Can Be Performed on Structures	408
10.3	Initializing Structures	409
10.4	Accessing Structure Members	409
10.5	Using Structures with Functions	411
10.6	typedef	411
10.7	Example: High-Performance Card Shuffling and Dealing Simulation	412
10.8	Unions	415
10.8.1	Union Declarations	415
10.8.2	Operations That Can Be Performed on Unions	415
10.8.3	Initializing Unions in Declarations	416
10.8.4	Demonstrating Unions	416
10.9	Bitwise Operators	417
10.9.1	Displaying an Unsigned Integer in Bits	418
10.9.2	Making Function displayBits More Scalable and Portable	420
10.9.3	Using the Bitwise AND, Inclusive OR, Exclusive OR and Complement Operators	420

10.9.4	Using the Bitwise Left- and Right-Shift Operators	423
10.9.5	Bitwise Assignment Operators	425
10.10	Bit Fields	426
10.11	Enumeration Constants	429
10.12	Secure C Programming	431
11	C File Processing	441
11.1	Introduction	442
11.2	Files and Streams	442
11.3	Creating a Sequential-Access File	443
11.4	Reading Data from a Sequential-Access File	448
11.5	Random-Access Files	452
11.6	Creating a Random-Access File	453
11.7	Writing Data Randomly to a Random-Access File	455
11.8	Reading Data from a Random-Access File	458
11.9	Case Study: Transaction-Processing Program	459
11.10	Secure C Programming	465
12	C Data Structures	476
12.1	Introduction	477
12.2	Self-Referential Structures	478
12.3	Dynamic Memory Allocation	478
12.4	Linked Lists	479
12.4.1	Function insert	485
12.4.2	Function delete	487
12.4.3	Function printList	488
12.5	Stacks	488
12.5.1	Function push	492
12.5.2	Function pop	492
12.5.3	Applications of Stacks	493
12.6	Queues	494
12.6.1	Function enqueue	498
12.6.2	Function dequeue	499
12.7	Trees	500
12.7.1	Function insertNode	504
12.7.2	Traversals: Functions inOrder, preOrder and postOrder	504
12.7.3	Duplicate Elimination	505
12.7.4	Binary Tree Search	505
12.7.5	Other Binary Tree Operations	505
12.8	Secure C Programming	506
13	C Preprocessor	517
13.1	Introduction	518
13.2	#include Preprocessor Directive	518

13.3	#define Preprocessor Directive: Symbolic Constants	519
13.4	#define Preprocessor Directive: Macros	519
13.5	Conditional Compilation	521
13.6	#error and #pragma Preprocessor Directives	522
13.7	# and ## Operators	523
13.8	Line Numbers	523
13.9	Predefined Symbolic Constants	523
13.10	Assertions	524
13.11	Secure C Programming	524

14 Other C Topics **529**

14.1	Introduction	530
14.2	Redirecting I/O	530
14.3	Variable-Length Argument Lists	531
14.4	Using Command-Line Arguments	533
14.5	Notes on Compiling Multiple-Source-File Programs	534
14.6	Program Termination with exit and atexit	536
14.7	Suffixes for Integer and Floating-Point Literals	537
14.8	Signal Handling	538
14.9	Dynamic Memory Allocation: Functions calloc and realloc	540
14.10	Unconditional Branching with goto	541

15 C++ as a Better C; Introducing Object Technology **547**

15.1	Introduction	548
15.2	C++	548
15.3	A Simple Program: Adding Two Integers	549
15.4	C++ Standard Library	551
15.5	Header Files	552
15.6	Inline Functions	554
15.7	References and Reference Parameters	556
15.8	Empty Parameter Lists	561
15.9	Default Arguments	561
15.10	Unary Scope Resolution Operator	563
15.11	Function Overloading	564
15.12	Function Templates	567
15.13	Introduction to C++ Standard Library Class Template vector	570
15.14	Introduction to Object Technology and the UML	576
15.15	Wrap-Up	579

16 Introduction to Classes, Objects and Strings **586**

16.1	Introduction	587
16.2	Defining a Class with a Member Function	587

16.3	Defining a Member Function with a Parameter	590
16.4	Data Members, <i>set</i> Functions and <i>get</i> Functions	593
16.5	Initializing Objects with Constructors	599
16.6	Placing a Class in a Separate File for Reusability	603
16.7	Separating Interface from Implementation	606
16.8	Validating Data with <i>set</i> Functions	612
16.9	Wrap-Up	617

17 Classes: A Deeper Look, Part I 623

17.1	Introduction	624
17.2	Time Class Case Study	625
17.3	Class Scope and Accessing Class Members	632
17.4	Separating Interface from Implementation	633
17.5	Access Functions and Utility Functions	634
17.6	Time Class Case Study: Constructors with Default Arguments	637
17.7	Destructors	642
17.8	When Constructors and Destructors Are Called	643
17.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member	646
17.10	Default Memberwise Assignment	649
17.11	Wrap-Up	652

18 Classes: A Deeper Look, Part 2 658

18.1	Introduction	659
18.2	const (Constant) Objects and const Member Functions	659
18.3	Composition: Objects as Members of Classes	667
18.4	friend Functions and friend Classes	673
18.5	Using the this Pointer	675
18.6	static Class Members	680
18.7	Proxy Classes	685
18.8	Wrap-Up	689

19 Operator Overloading; Class string 695

19.1	Introduction	696
19.2	Using the Overloaded Operators of Standard Library Class <code>string</code>	697
19.3	Fundamentals of Operator Overloading	700
19.4	Overloading Binary Operators	701
19.5	Overloading the Binary Stream Insertion and Stream Extraction Operators	702
19.6	Overloading Unary Operators	706
19.7	Overloading the Unary Prefix and Postfix ++ and -- Operators	707
19.8	Case Study: A Date Class	708
19.9	Dynamic Memory Management	713

19.10	Case Study: Array Class	715
19.10.1	Using the Array Class	716
19.10.2	Array Class Definition	719
19.11	Operators as Member Functions vs. Non-Member Functions	727
19.12	Converting between Types	727
19.13	explicit Constructors	729
19.14	Building a String Class	731
19.15	Wrap-Up	732

20 Object-Oriented Programming: Inheritance 743

20.1	Introduction	744
20.2	Base Classes and Derived Classes	744
20.3	protected Members	747
20.4	Relationship between Base Classes and Derived Classes	747
20.4.1	Creating and Using a CommissionEmployee Class	748
20.4.2	Creating a BasePlusCommissionEmployee Class Without Using Inheritance	752
20.4.3	Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy	758
20.4.4	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data	763
20.4.5	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data	766
20.5	Constructors and Destructors in Derived Classes	771
20.6	public, protected and private Inheritance	771
20.7	Software Engineering with Inheritance	772
20.8	Wrap-Up	773

21 Object-Oriented Programming: Polymorphism 778

21.1	Introduction	779
21.2	Introduction to Polymorphism: Polymorphic Video Game	780
21.3	Relationships Among Objects in an Inheritance Hierarchy	780
21.3.1	Invoking Base-Class Functions from Derived-Class Objects	781
21.3.2	Aiming Derived-Class Pointers at Base-Class Objects	784
21.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	785
21.3.4	Virtual Functions	787
21.4	Type Fields and switch Statements	793
21.5	Abstract Classes and Pure virtual Functions	793
21.6	Case Study: Payroll System Using Polymorphism	795
21.6.1	Creating Abstract Base Class Employee	796
21.6.2	Creating Concrete Derived Class SalariedEmployee	800
21.6.3	Creating Concrete Derived Class CommissionEmployee	802
21.6.4	Creating Indirect Concrete Derived Class BasePlusCommissionEmployee	804
21.6.5	Demonstrating Polymorphic Processing	806

21.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	810
21.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>type_info</code>	813
21.9	Virtual Destructors	817
21.10	Wrap-Up	817
22	Templates	823
22.1	Introduction	824
22.2	Function Templates	824
22.3	Overloading Function Templates	827
22.4	Class Templates	828
22.5	Nontype Parameters and Default Types for Class Templates	834
22.6	Wrap-Up	835
23	Stream Input/Output	839
23.1	Introduction	840
23.2	Streams	841
23.2.1	Classic Streams vs. Standard Streams	841
23.2.2	<code>iostream</code> Library Headers	842
23.2.3	Stream Input/Output Classes and Objects	842
23.3	Stream Output	845
23.3.1	Output of <code>char *</code> Variables	845
23.3.2	Character Output Using Member Function <code>put</code>	845
23.4	Stream Input	846
23.4.1	<code>get</code> and <code>getline</code> Member Functions	846
23.4.2	<code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	849
23.4.3	Type-Safe I/O	849
23.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	849
23.6	Introduction to Stream Manipulators	850
23.6.1	Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	851
23.6.2	Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	851
23.6.3	Field Width (<code>width</code> , <code>setw</code>)	853
23.6.4	User-Defined Output Stream Manipulators	854
23.7	Stream Format States and Stream Manipulators	856
23.7.1	Trailing Zeros and Decimal Points (<code>showpoint</code>)	856
23.7.2	Justification (<code>left</code> , <code>right</code> and <code>internal</code>)	857
23.7.3	Padding (<code>fill</code> , <code>setfill</code>)	859
23.7.4	Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>)	860
23.7.5	Floating-Point Numbers; Scientific and Fixed Notation (<code>scientific</code> , <code>fixed</code>)	861
23.7.6	Uppercase/Lowercase Control (<code>uppercase</code>)	862
23.7.7	Specifying Boolean Format (<code>boolalpha</code>)	862
23.7.8	Setting and Resetting the Format State via Member Function flags	863

23.8	Stream Error States	864
23.9	Tying an Output Stream to an Input Stream	866
23.10	Wrap-Up	867

24 Exception Handling: A Deeper Look **876**

24.1	Introduction	877
24.2	Example: Handling an Attempt to Divide by Zero	877
24.3	When to Use Exception Handling	883
24.4	Rethrowing an Exception	884
24.5	Processing Unexpected Exceptions	885
24.6	Stack Unwinding	886
24.7	Constructors, Destructors and Exception Handling	888
24.8	Exceptions and Inheritance	888
24.9	Processing new Failures	889
24.10	Class <code>unique_ptr</code> and Dynamic Memory Allocation	892
24.11	Standard Library Exception Hierarchy	894
24.12	Wrap-Up	896

A Operator Precedence Charts **902**

B ASCII Character Set **906**

C Number Systems **907**

C.1	Introduction	908
C.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	911
C.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	912
C.4	Converting from Binary, Octal or Hexadecimal to Decimal	912
C.5	Converting from Decimal to Binary, Octal or Hexadecimal	913
C.6	Negative Binary Numbers: Two's Complement Notation	915

D Game Programming: Solving Sudoku **920**

D.1	Introduction	920
D.2	Deitel Sudoku Resource Center	921
D.3	Solution Strategies	921
D.4	Programming Sudoku Puzzle Solvers	925
D.5	Generating New Sudoku Puzzles	926
D.6	Conclusion	928

Appendices on the Web **929**

Index **930**

Appendices E through H are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).

E Sorting: A Deeper Look

F Introduction to the New C Standard

G Using the Visual Studio Debugger

H Using the GNU Debugger

A Operator Precedence Charts

B ASCII Character Set

C Number Systems

D Game Programming: Solving Sudoku

Appendices on the Web

Index

877	Introduction	24.1
878	Example: Handling	24.2
883	When to Use Exception Handling	24.3
884	Rethrowing an Exception	24.4
888	Processing Unchecked	24.5
888	Stack Unwinding	24.6
888	Constructors, Destructors and Exception Handling	24.7
888	Exceptions and Inheritance	24.8
889	Processing new Failures	24.9
892	Class <code>nullptr</code> and Dynamic Memory Allocation	24.10
894	Standard Library Exception Hierarchy	24.11
896	Wrap-Up	24.12
897	Introduction	25.1
898	Streams	25.2
900	25.2.1 Classic Streams vs. <code>std::basic_ostream</code>	25.2.1
900	25.2.2 <code>std::basic_istream</code> and <code>std::basic_ostream</code>	25.2.2
900	25.2.3 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.2.3
907	25.3 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3
908	25.3.1 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.1
911	25.3.2 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.2
912	25.3.3 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.3
912	25.3.4 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.4
913	25.3.5 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.5
915	25.3.6 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.6
915	25.3.7 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.7
920	25.3.8 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.8
920	25.3.9 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.9
921	25.3.10 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.10
921	25.3.11 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.11
922	25.3.12 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.12
926	25.3.13 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.13
928	25.3.14 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.14
928	25.3.15 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.15
928	25.3.16 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.16
928	25.3.17 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.17
928	25.3.18 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.18
928	25.3.19 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.19
928	25.3.20 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.20
928	25.3.21 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.21
928	25.3.22 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.22
928	25.3.23 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.23
928	25.3.24 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.24
928	25.3.25 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.25
928	25.3.26 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.26
928	25.3.27 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.27
928	25.3.28 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.28
928	25.3.29 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.29
928	25.3.30 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.30
928	25.3.31 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.31
928	25.3.32 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.32
928	25.3.33 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.33
928	25.3.34 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.34
928	25.3.35 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.35
928	25.3.36 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.36
928	25.3.37 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.37
928	25.3.38 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.38
928	25.3.39 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.39
928	25.3.40 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.40
928	25.3.41 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.41
928	25.3.42 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.42
928	25.3.43 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.43
928	25.3.44 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.44
928	25.3.45 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.45
928	25.3.46 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.46
928	25.3.47 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.47
928	25.3.48 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.48
928	25.3.49 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.49
928	25.3.50 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.50
928	25.3.51 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.51
928	25.3.52 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.52
928	25.3.53 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.53
928	25.3.54 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.54
928	25.3.55 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.55
928	25.3.56 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.56
928	25.3.57 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.57
928	25.3.58 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.58
928	25.3.59 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.59
928	25.3.60 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.60
928	25.3.61 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.61
928	25.3.62 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.62
928	25.3.63 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.63
928	25.3.64 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.64
928	25.3.65 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.65
928	25.3.66 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.66
928	25.3.67 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.67
928	25.3.68 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.68
928	25.3.69 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.69
928	25.3.70 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.70
928	25.3.71 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.71
928	25.3.72 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.72
928	25.3.73 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.73
928	25.3.74 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.74
928	25.3.75 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.75
928	25.3.76 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.76
928	25.3.77 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.77
928	25.3.78 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.78
928	25.3.79 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.79
928	25.3.80 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.80
928	25.3.81 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.81
928	25.3.82 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.82
928	25.3.83 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.83
928	25.3.84 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.84
928	25.3.85 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.85
928	25.3.86 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.86
928	25.3.87 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.87
928	25.3.88 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.88
928	25.3.89 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.89
928	25.3.90 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.90
928	25.3.91 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.91
928	25.3.92 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.92
928	25.3.93 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.93
928	25.3.94 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.94
928	25.3.95 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.95
928	25.3.96 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.96
928	25.3.97 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.97
928	25.3.98 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.98
928	25.3.99 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.3.99
928	25.4 <code>std::basic_ostream</code> and <code>std::basic_istream</code>	25.4