
Contents

Preface

Chapter 1	Introduction to Computers and C++ Programming	1
1.1	Introduction	2
1.2	What Is a Computer?	5
1.3	Computer Organization	5
1.4	Evolution of Operating Systems	6
1.5	Personal Computing, Distributed Computing, and Client/Server Computing	7
1.6	Machine Languages, Assembly Languages, and High-level Languages	7
1.7	The History of C++	9
1.8	C++ Class Libraries and the C Standard Library	10
1.9	Concurrent C++	11
1.10	Other High-level Languages	11
1.11	Structured Programming	11
1.12	Basics of a Typical C++ Environment	12
1.13	General Notes About C++ and this Book	15
1.14	Introduction to C++ Programming	16
1.15	A Simple Program: Printing a Line of Text	16
1.16	Another Simple Program: Adding Two Integers	20
1.17	Memory Concepts	24
1.18	Arithmetic	25
1.19	Decision Making: Equality and Relational Operators	28
1.20	Thinking About Objects	32
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	

Chapter 2	Control Structures	51
2.1	Introduction	52
2.2	Algorithms	53
2.3	Pseudocode	53
2.4	Control Structures	54
2.5	The If Selection Structure	56
2.6	The If/Else Selection Structure	58
2.7	The While Repetition Structure	62
2.8	Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)	63
2.9	Formulating Algorithms with Top-down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)	65
2.10	Formulating Algorithms with Top-down, Stepwise Refinement: Case Study 3 (Nested Control Structures)	72
2.11	Assignment Operators	75
2.12	Increment and Decrement Operators	77
2.13	Essentials of Counter-Controlled Repetition	80
2.14	The For Repetition Structure	82
2.15	Examples Using the For Structure	85
2.16	The Switch Multiple-Selection Structure	89
2.17	The Do/While Repetition Structure	96
2.18	The Break and Continue Statements	97
2.19	Logical Operators	99
2.20	Confusing Equality (==) and Assignment (=) Operators	103
2.21	Structured Programming Summary	104
2.22	Thinking About Objects: Identifying the Objects in a Problem	110
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 3	Functions	135
3.1	Introduction	136
3.2	Program Modules in C++	137
3.3	Math Library Functions	138
3.4	Functions	139
3.5	Function Definitions	140
3.6	Function Prototypes	143
3.7	Header Files	147
3.8	Random Number Generation	148
3.9	Example: A Game of Chance	153
3.10	Storage Classes	156
3.11	Scope Rules	159

3.12	Recursion	162
3.13	Example Using Recursion: The Fibonacci Series	165
3.14	Recursion vs. Iteration	169
3.15	Functions with Empty Parameter Lists	171
3.16	Inline Functions	172
3.17	References and Reference Parameters	172
3.18	Default Arguments	177
3.19	Unary Scope Resolution Operator	178
3.20	Function Overloading	179
3.21	Function Templates	181
3.22	Thinking About Objects: Identifying an Object's Attributes	182
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 4	Arrays	211
4.1	Introduction	212
4.2	Arrays	212
4.3	Declaring Arrays	214
4.4	Examples Using Arrays	215
4.5	Passing Arrays to Functions	228
4.6	Sorting Arrays	231
4.7	Case Study: Computing Mean, Median, and Mode Using Arrays	233
4.8	Searching Arrays: Linear Search and Binary Search	236
4.9	Multiple-Subscripted Arrays	239
4.10	Thinking About Objects: Identifying an Object's Behaviors	248
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises • Recursion Exercises</i>	
Chapter 5	Pointers and Strings	267
5.1	Introduction	268
5.2	Pointer Variable Declarations and Initialization	269
5.3	Pointer Operators	270
5.4	Calling Functions by Reference	272
5.5	Using the const Qualifier with Pointers	277
5.6	Bubble Sort Using Call-by-Reference	282
5.7	Pointer Expressions and Pointer Arithmetic	287

5.8	The Relationship Between Pointers and Arrays	290
5.9	Arrays of Pointers	294
5.10	Case Study: A Card Shuffling and Dealing Simulation	295
5.11	Function Pointers	300
5.12	Introduction to Character and String Processing	304
	5.12.1 Fundamentals of Characters and Strings	304
	5.12.2 String Manipulation Functions of the String Handling Library	304
5.13	Thinking About Objects: Interactions Among Objects	313
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portabil- ity Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises • Special Section: Building Your Own Computer • More Pointer Exercises • String Manipulation Exercises • Special Section: Advanced String Manipulation Exercises • A Challenging String Manipulation Project</i>	
Chapter 6	Classes and Data Abstraction	345
6.1	Introduction	346
6.2	Structured Definitions	347
6.3	Accessing Members of Structures	348
6.4	Implementating a User-Defined Type Time With a Struct	349
6.5	Implementating a Time Abstract Data Type With a Class	351
6.6	Class Scope and Accessing Class Members	357
6.7	Separating Interface from Implementation	357
6.8	Controlling Access to Members	362
6.9	Access Functions and Utility Functions	364
6.10	Initializing Class Objects: Constructors	366
6.11	Using Default Arguments with Constructors	368
6.12	Using Destructors	371
6.13	When Destructors and Constructors are Called	372
6.14	Using Data Members and Member Functions	373
6.15	A Subtle Trap: Returning a Reference to Private Data Member	379
6.16	Assignment by Default Memberwise Copy	381
6.17	Software Reusability	383
6.18	Thinking About Objects: Programming the Classes for the Elevator Simulator	383
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	

Chapter 7	Classes: Part II	395
7.1	Introduction	396
7.2	Constant Objects and const Member Functions	396
7.3	Composition: Classes as Members of Other Classes	403
7.4	Friend Functions and Friend Classes	405
7.5	Using the This Pointer	409
7.6	Dynamic Memory Allocation with Operators New and Delete	414
7.7	Static Class Members	415
7.8	Data Abstraction and Information Hiding	420
	7.8.1 Example: Array Abstract Data Type	421
	7.8.2 Example: String Abstract Data Type	422
	7.8.3 Example: Queue Abstract Data Type	423
7.9	Container Classes and Iterators	423
7.10	Thinking About Objects: Using Composition and Dynamic Object Management in the Elevator Simulator	424
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 8	Operator Overloading	431
8.1	Introduction	432
8.2	Fundamentals of Operator Overloading	433
8.3	Restrictions on Operator Overloading	434
8.4	Operator Functions as Class Members vs. as Friend Functions	436
8.5	Overloading Stream-Insertion and Stream-Extraction Operators	436
8.6	Overloading Unary Operators	440
8.7	Overloading Binary Operators	441
8.8	Case Study: An Array Class	441
8.9	Converting Between Types	452
8.10	Case Study: A String Class	452
8.11	Overloading ++ and --	464
8.12	Case Study: A Date Class	465
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 9	Inheritance	483
9.1	Introduction	484
9.2	Base Classes and Derived Classes	486
9.3	Protected Members	486

9.4	Casting Base-Class Pointers to Derived-Class Pointers	486
9.5	Using Member Functions	486
9.6	Redefining Base-Class Members in a Derived Class	486
9.7	Public, Protected, and Private Base Classes	486
9.8	Direct Base Classes and Indirect Base Classes	498
9.9	Using Constructors and Destructors in Derived Classes	499
9.10	Implicit Derived-Class Object to Base-Class Object Conversion	501
9.11	Software Engineering with Inheritance	503
9.12	Composition vs. Inheritance	505
9.13	“Uses A” and “Knows A” Relationships	505
9.14	Case Study: Point, Circle, Cylinder	505
9.15	Multiple Inheritance	511
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tip • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 10	Virtual Functions and Polymorphism	525
10.1	Introduction	526
10.2	Type Fields and Switch Statements	526
10.3	Virtual Functions	527
10.4	Abstract Base Classes and Concrete Classes	527
10.5	Polymorphism	529
10.6	Case Study: A Payroll System Using Polymorphism	529
10.7	New Classes and Dynamic Binding	540
10.8	Virtual Destructors	540
10.9	Case Study: Inheriting Interface and Implementation	542
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 11	Stream Input/Output	555
11.1	Introduction	557
11.2	Streams	557
11.2.1	Iostream Library Header Files	558
11.2.2	Stream Input/Output Classes and Objects	558
11.3	Stream Output	560
11.3.1	Stream-Insertion Operator	560
11.3.2	Concatenating Stream Insertion/Extraction Operators	562
11.3.3	Output of Char* Variables	563

	11.3.4	Character Output with the Put Member Function; Concatenating Puts	563
11.4		Stream Input	564
	11.4.1	Stream-Extraction Operator	564
	11.4.2	Get and Getline Member Functions	567
	11.4.3	Other Istream Member Functions (peek, putback, ignore)	569
	11.4.4	Type-Safe I/O	570
11.5		Unformatted I/O with Read, Gcount and Write	570
11.6		Stream Manipulators	570
	11.6.1	Integral Stream Base: Dec, Oct, Hex and Setbase Stream Manipulators	570
	11.6.2	Floating-Point Precision (precision, setprecision)	571
	11.6.3	Field Width (setw, width)	572
	11.6.4	User-Defined Manipulators	574
11.7		Stream Format States	574
	11.7.1	Format State Flags (setf, unsetf, flags)	576
	11.7.2	Trailing Zeros and Decimal Points (ios::showpoint)	576
	11.7.3	Justification (ios::left, ios::right, ios::internal)	577
	11.7.4	Padding (fill, setfill)	579
	11.7.5	Integral Stream Base (ios::dec, ios::oct, ios::hex, ios::showbase)	579
	11.7.6	Floating-Point Numbers; Scientific Notation (ios::scientific, ios::fixed)	581
	11.7.7	Uppercase/Lowercase Control (ios::uppercase)	582
	11.7.8	Setting and Resetting the Format Flags (setiosflags, resetiosflags)	582
11.8		Stream Error States	583
11.9		I/O of User-Defined Types	585
11.10		Tying an Output Stream to an Input Stream	587
		<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 12		Templates	603
	12.1	Introduction	604
	12.2	Function Templates	605
	12.3	Overloading Template Functions	607
	12.4	Class Templates	608
	12.5	Class Templates and Non-Type Parameters	612

12.6	Templates and Inheritance	613
12.7	Templates and Friends	613
12.8	Templates and Static Members	615
	<i>Summary • Terminology • Common Programming Errors • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 13	Exception Handling	621
13.1	Introduction	622
13.2	When Exception Handling Should Be Used	625
13.3	Other Error-Handling Techniques	625
13.4	The Basics of C++ Exception Handling	626
13.5	A Simple Exception Handling Example: Divide by Zero	627
13.6	Try Blocks	629
13.7	Throwing an Exception	630
13.8	Catching an Exception	631
13.9	Rethrowing an Exception	634
13.10	Throwing a Conditional Expression	635
13.11	Exception Specifications	635
13.12	Processing Unexpected Exceptions	636
13.13	Constructors, Destructors, and Exception Handling	637
13.14	Exceptions and Inheritance	637
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 14	File Processing and String Stream I/O	649
14.1	Introduction	650
14.2	The Data Hierarchy	650
14.3	Files and Streams	652
14.4	Creating a Sequential Access File	653
14.5	Reading Data from a Sequential Access File	658
14.6	Updating Sequential Access Files	662
14.7	Random Access Files	663
14.8	Creating a Randomly Accessed File	664
14.9	Writing Data Randomly to a Random Access File	666
14.10	Reading Data Sequentially from a Random Access File	666
14.11	Example: A Transaction Processing Program	668
14.12	String Stream Processing	675
14.13	Input/Output of Objects	679

	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tip • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 15	Data Structures	691
15.1	Introduction	692
15.2	Self-Referential Classes	693
15.3	Dynamic Memory Allocation	694
15.4	Linked Lists	695
15.5	Stacks	708
15.6	Queues	713
15.7	Trees	716
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Performance Tips • Portability Tip • Self-Review Exercises • Answers to Self-Review Exercises • Exercises • Special Section: Building Your Own Compiler</i>	
Chapter 16	Bits, Characters, Strings and Structures	749
16.1	Introduction	750
16.2	Structure Definitions	750
16.3	Initializing Structures	753
16.4	Using Structures with Functions	753
16.5	Typedef	753
16.6	Example: High-Performance Card Shuffling and Dealing Simulation	754
16.7	Bitwise Operators	756
16.8	Bit Fields	764
16.9	Character Handling Library	767
16.10	String Conversion Functions	772
16.11	Search Functions of the String Handling Library	777
16.12	Memory Functions of the String Handling Library	781
16.13	Other Functions of the String Handling Library	786
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practices • Portability Tips • Self-Review Exercises • Answers to Self-Review Exercises • Exercises • Special Section: A Compendium of More Advanced String Manipulation Exercises</i>	
Chapter 17	The Preprocessor	801
17.1	Introduction	802
17.2	The #include Preprocessor Directive	802

17.3	The #define Preprocessor Directive: Symbolic Constants	803
17.4	The #define Preprocessor Directive: Macros	803
17.5	Conditional Compilation	805
17.6	The #error and #pragma Preprocessor Directives	806
17.7	The # and ## Operators	807
17.8	Line Numbers	807
17.9	Predefined Symbolic Constants	808
17.10	Assertions	808
	<i>Summary • Terminology • Common Programming Errors • Good Programming Practice • Performance Tip • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Chapter 18	Other Topics	815
18.1	Introduction	816
18.2	Redirecting Input/Output on UNIX and DOS Systems	816
18.3	Variable-Length Argument Lists	817
18.4	Using Command-Line Arguments	818
18.5	Notes on Compiling Multiple-Source-File Programs	820
18.6	Program Termination with Exit and Atexit	822
18.7	The Volatile Type Qualifier	824
18.8	Suffixes for Integer and Floating-Point Constants	824
18.9	Signal Handling	825
18.10	Dynamic Memory Allocation: Functions Calloc and Realloc	825
18.11	The Unconditional Branch: Goto	827
18.12	Unions	829
18.13	Linkage Specifications	831
18.14	Closing Remarks	833
	<i>Summary • Terminology • Common Programming Error • Portability Tips • Performance Tips • Software Engineering Observations • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	
Appendix A	C++ Syntax	840
A.1	Keywords	840
A.2	Lexical conventions	841
A.3	Basic concepts	844
A.4	Expressions	844
A.5	Statements	847
A.6	Declarations	848
A.7	Declarators	851
A.8	Classes	852
A.9	Derived classes	853

A.10	Special member functions	853
A.11	Overloading	854
A.12	Templates	854
A.13	Exception handling	855
Appendix B	Standard Library	856
B.1	Errors <code><errno.h></code>	856
B.2	Common Definitions <code><stddef.h></code>	856
B.3	Diagnostics <code><assert.h></code>	857
B.4	Character Handling <code><ctype.h></code>	857
B.5	Localization <code><locale.h></code>	858
B.6	Mathematics <code><math.h></code>	861
B.7	Nonlocal Jumps <code><setjmp.h></code>	863
B.8	Signal Handling <code><signal.h></code>	863
B.9	Variable Arguments <code><stdarg.h></code>	865
B.10	Input/Output <code><stdio.h></code>	865
B.11	General Utilities <code><stdlib.h></code>	873
B.12	String Handling <code><string.h></code>	879
B.13	Date and Time <code><time.h></code>	882
B.14	Implementation Limits:	
	<code><limits.h></code>	885
	<code><float.h></code>	886
Appendix C	Operator Precedence Chart	888
Appendix D	ASCII Character Set	889
Appendix E	Number Systems	891
E.1	Introduction	892
E.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	895
E.3	Converting from Octal Numbers and Hexadecimal Numbers to Binary Numbers	896
E.4	Converting from Binary, Octal, or Hexadecimal to Decimal	896
E.5	Converting from Decimal to Binary, Octal, or Hexadecimal	897
E.6	Negative Binary Numbers: Two's Complement Notation	899
	<i>Summary • Terminology • Self-Review Exercises • Answers to Self-Review Exercises • Exercises</i>	

Appendix F	Resources	905
Bibliography		909
Index		919

Illustrations

Chapter 1	Introduction to Computers and C++ Programming	
1.1	A typical C++ environment.	13
1.2	Text printing program.	16
1.3	Some common escape sequences.	18
1.4	Printing on one line with separate statements using <code>cout</code> .	19
1.5	Printing on multiple lines with a single statement using <code>cout</code> .	20
1.6	An addition program.	20
1.7	A memory location showing the name and value of a variable.	24
1.8	Memory locations after values for two variables have been input.	24
1.9	Memory locations after a calculation.	25
1.10	Arithmetic operators.	25
1.11	Precedence of arithmetic operators.	27
1.12	Order in which a second degree polynomial is evaluated.	29
1.13	Equality and relational operators.	29
1.14	Using equality and relational operators.	30
1.15	Precedence and associativity of the operators discussed so far.	32
Chapter 2	Control Structures	
2.1	Flowcharting C++'s sequence structure.	55
2.2	C++ keywords.	56
2.3	Flowcharting the single-selection <code>if</code> structure.	56
2.4	Flowcharting the double-selection <code>if/else</code> structure.	59
2.5	Flowcharting the <code>while</code> repetition structure.	63
2.6	Pseudocode algorithm that uses counter-controlled repetition to solve the class average problem.	64

2.7	C++ program and sample execution for the class average problem with counter-controlled repetition.	64
2.8	Pseudocode algorithm that uses sentinel-controlled repetition to solve the class average problem.	68
2.9	C++ program and sample execution for the class average problem with sentinel-controlled repetition.	68
2.10	Pseudocode for examination results problem.	74
2.11	C++ program and sample executions for examination results problem.	75
2.12	Arithmetic assignment operators.	76
2.13	The increment and decrement operators.	77
2.14	The difference between preincrementing and postincrementing.	78
2.15	Precedence of the operators encountered so far in the text.	79
2.16	Counter-controlled repetition.	80
2.17	Counter-controlled repetition with the for structure.	82
2.18	Components of a typical for header.	83
2.19	Flowcharting a typical for structure.	83
2.20	Summation with for .	86
2.21	Calculating compound interest with for .	88
2.22	An example using switch .	91
2.23	The switch multiple-selection structure.	94
2.24	Using the do/while structure.	97
2.25	The do/while repetition structure.	97
2.26	Using the break statement in a for structure.	98
2.27	Using the continue statement in a for structure.	99
2.28	Truth table for the && (logical AND) operator.	100
2.29	Truth table for the logical OR () operator.	101
2.30	Truth table for operator ! (logical negation).	102
2.31	Operator precedence and associativity.	102
2.32	C++'s single-entry/single-exit sequence, selection, and repetition structures.	104
2.33	Rules for forming structured programs.	106
2.34	The simplest flowchart.	106
2.35	Repeatedly applying rule 2 of Fig. 2.33 to the simplest flowchart.	106
2.36	Applying rule 3 of Fig. 2.33 to the simplest flowchart.	107
2.37	Stacked building blocks, nested building blocks, and overlapped building blocks.	108
2.38	An unstructured flowchart.	109
Chapter 3 Functions		
3.1	Hierarchical boss function/worker function relationship.	138
3.2	Commonly used math library functions.	139

3.3	Using a programmer-defined function.	141
3.4	Programmer-defined maximum function.	143
3.5	Promotion hierarchy for built-in data types.	146
3.6	The standard library header files.	147
3.7	Shifted, scaled integers produced by <code>1 + rand() % 6</code> .	149
3.8	Rolling a six-sided die 6000 times.	150
3.9	Randomizing the die-rolling program.	151
3.10	Program to simulate the game of craps.	154
3.11	Sample runs for the game of craps.	155
3.12	A scoping example.	160
3.13	Recursive evaluation of 5!.	164
3.14	Calculating factorials with a recursive function.	164
3.15	Recursively generating Fibonacci numbers.	166
3.16	Set of recursive calls to function fibonacci .	167
3.17	Summary of recursion examples and exercises in the text.	170
3.18	Two ways to declare and use functions that take no arguments.	171
3.19	Using an inline function to calculate the volume of a cube.	173
3.20	An example of call-by-reference.	174
3.21	Attempting to use an uninitialized reference.	176
3.22	Using an initialized reference.	176
3.23	Using default arguments.	177
3.24	Using the unary scope resolution operator.	179
3.25	Using overloaded functions.	180
3.26	Name mangling to enable type-safe linkage.	180
3.27	Using template functions.	182
3.28	The Towers of Hanoi for the case with four disks.	205
Chapter 4	Arrays	
4.1	A 12-element array.	213
4.2	Operator precedence.	214
4.3	Initializing the elements of an array to zeros.	215
4.4	Initializing the elements of an array with a declaration.	216
4.5	Generating the values to be placed into elements of an array.	217
4.6	A const object must be initialized.	218
4.7	Correctly initializing and using a constant variable.	218
4.8	Computing the sum of the elements of an array.	219
4.9	A simple student poll analysis program.	220
4.10	A program that prints histograms.	222
4.11	Dice-rolling program using arrays instead of switch .	223
4.12	Treating character arrays as strings.	225

4.13	Comparing static array initialization and automatic array initialization.	226
4.14	Passing arrays and individual array elements to functions.	230
4.15	Demonstrating the const type qualifier.	231
4.16	Sorting an array with bubble sort.	233
4.17	Survey data analysis program.	234
4.18	Sample run for the survey data analysis program.	237
4.19	Linear search of an array.	238
4.20	Binary search of a sorted array.	240
4.21	A double-subscripted array with three rows and four columns.	243
4.22	Initializing multidimensional arrays.	244
4.23	Example of using double-subscripted arrays.	246
4.24	The 36 possible outcomes of rolling two dice.	258
4.25	The eight possible moves of the knight.	261
4.26	The 22 squares eliminated by placing a queen in the upper left corner.	264
Chapter 5 Pointers and Strings		
5.1	Directly and indirectly referencing a variable.	270
5.2	Graphical representation of a pointer pointing to an integer variable in memory.	271
5.3	Representation of y and yPtr in memory.	271
5.4	The & and * pointer operators.	272
5.5	Operator precedence.	272
5.6	Cube a variable using call-by-value.	274
5.7	Cube a variable using call-by-reference with a pointer argument.	274
5.8	Analysis of a typical call-by-value.	276
5.9	Analysis of a typical call-by-reference with a pointer argument.	277
5.10	Converting a string to uppercase.	279
5.11	Printing a string one character at a time using a non-constant pointer to constant data.	280
5.12	Attempting to modify data through a non-constant pointer to constant data.	281
5.13	Attempting to modify a constant pointer to non-constant data.	282
5.14	Attempting to modify a constant pointer to constant data.	283
5.15	Bubble sort with call-by-reference.	284
5.16	The sizeof operator when applied to an array name returns the number of bytes in the array.	286
5.17	Using the sizeof operator to determine standard data type sizes.	287

5.18	The array <code>v</code> and a pointer variable <code>vPtr</code> that points to <code>v</code> .	288
5.19	The pointer <code>vPtr</code> after pointer arithmetic.	289
5.20	Using four methods of referencing array elements.	293
5.21	Copying a string using array notation and pointer notation.	294
5.22	A graphical representation of the <code>suit</code> array.	295
5.23	Double-subscripted array representation of a deck of cards.	296
5.24	Card dealing program.	299
5.25	Sample run of card dealing program.	300
5.26	Multipurpose sorting program using function pointers.	301
5.27	The outputs of the bubble sort program in Fig. 5.26.	303
5.28	Demonstrating an array of pointers to functions.	304
5.29	The string manipulation functions of the string handling library.	308
5.30	Using <code>strcpy</code> and <code>strncpy</code> .	310
5.31	Using <code>strcat</code> and <code>strncat</code> .	310
5.32	Using <code>strcmp</code> and <code>strncmp</code> .	311
5.33	Using <code>strtok</code> .	313
5.34	Using <code>strlen</code> .	314
5.35	Unshuffled <code>deck</code> array.	327
5.36	Sample shuffled <code>deck</code> array.	327
5.37	Simpletron Machine Language (SML) operation codes.	329
5.38	A sample dump.	333
5.39	The letters of the alphabet as expressed in international Morse code.	343
Chapter 6	Classes and Data Abstraction	
6.1	Create a structure, set its members, and print it.	350
6.2	Simple definition of <code>class Time</code> .	351
6.3	Abstract data type <code>Time</code> implementation as a class.	353
6.4	Accessing an object's data members and member functions through the object's name, through a reference, and through a pointer to the object.	358
6.5	<code>Time</code> class header file.	359
6.5	<code>Time</code> class member function definitions source file.	360
6.5	Driver program for <code>Time</code> class.	361
6.6	Erroneous attempt to access private members of a class.	363
6.7	Using a utility function.	365
6.8	Using a constructor with default arguments.	368
6.9	Demonstrating the order in which constructors and destructors are called.	373
6.10	Declaration of the <code>Time</code> class.	376
6.10	Member function definitions for <code>Time</code> class.	377

6.10	Using set and get functions.	378
6.11	Returning a reference to a private data member.	380
6.12	Assigning one object to another with default memberwise copy.	382
Chapter 7	Classes: Part II	
7.1	Using a Time class with const objects and const member functions.	398
7.2	Using a member initializer to initialize a constant of a built-in data type.	401
7.3	Erroneous attempt to initialize a constant of a built-in data type by assignment.	402
7.4	Using member-object initializers.	403
7.5	Friends can access private members of a class.	408
7.6	Non-friend/non-member functions cannot access private class members.	409
7.7	Using the this pointer.	410
7.8	Chaining member function calls.	412
7.9	Using a static data member to maintain a count of the number of objects of a class.	416
Chapter 8	Operator Overloading	
8.1	Operators that can be overloaded.	434
8.2	Operators that cannot be overloaded.	435
8.3	User-defined stream-insertion and stream-extraction operators.	438
8.4	Definition of class Array .	439
8.4	Member function definitions for class Array .	443
8.4	Driver for class Array .	445
8.4	Output from driver for class Array .	447
8.5	Definition of a basic String class.	454
8.5	Member function definitions for class String .	454
8.5	Driver for testing class String .	458
8.5	Output from driver for testing class String .	460
8.6	Definition of class Date .	466
8.6	Member function definitions for class Date .	466
8.6	Driver for class Date .	469
8.6	Output from driver for class Date .	470
8.7	Definition of class Complex .	476
8.7	Member function definitions for class Complex .	476
8.7	Driver for class Complex .	477
8.7	Output from driver for class Complex .	478
8.8	A user-defined huge integer class.	478

Chapter 9	Inheritance	
9.1	Some simple inheritance examples.	486
9.2	An inheritance hierarchy for university community members.	487
9.3	A portion of a Shape class hierarchy.	487
9.4	Definition of class Point .	489
9.4	Member function definitions for class Point .	489
9.4	Definition of class Circle .	490
9.4	Member function definition for class Circle .	490
9.4	Casting base-class pointers to derived-class pointers.	491
9.5	Definition of class Employee .	494
9.5	Member function definitions for class Employee .	495
9.5	Definition of class HourlyWorker .	495
9.5	Member function definitions for class HourlyWorker .	496
9.5	Redefining a base-class member function in a derived class.	496
9.6	Summary of base-class member accessibility in a derived class.	498
9.7	Definition of class Point .	500
9.7	Member function definitions for class Point .	500
9.7	Definition of class Circle .	501
9.7	Member function definitions for class Circle .	501
9.7	Order in which base-class and derived-class constructors and destructors are called.	502
9.8	Definition of class Point .	506
9.8	Member functions for class Point .	506
9.8	Driver for class Point .	507
9.9	Definition of class Circle .	507
9.9	Member function definitions for class Circle .	508
9.9	Driver for class Circle .	509
9.10	Definition of class Cylinder .	510
9.10	Member function and friend function definitions for class Cylinder .	511
9.10	Driver for class Cylinder .	512
9.11	Definition of class Base1 .	513
9.11	Definition of class Base2 .	513
9.11	Definition of class Derived .	514
9.11	Member function definitions for class Derived .	514
9.11	Driver for the multiple inheritance example.	515
Chapter 10	Virtual Functions and Polymorphism	
10.1	Abstract base class Employee .	531
10.1	Member function definitions for abstract base class Employee .	531

10.1	Class Boss derived from abstract base class Employee .	533
10.1	Member function definitions for class Boss .	533
10.1	Class CommissionWorker derived from abstract base class Employee .	534
10.1	Member function definitions for class CommissionWorker .	535
10.1	Class PieceWorker derived from abstract base class Employee .	536
10.1	Member function definitions for class PieceWorker .	536
10.1	Class HourlyWorker derived from abstract base class Employee .	537
10.1	Member function definitions for class HourlyWorker .	537
10.1	Employee class derivation hierarchy that uses an abstract base class.	538
10.2	Definition of abstract base class Shape .	542
10.2	Definition of class Point .	542
10.2	Member function definitions for class Point .	543
10.2	Definition of class Circle .	543
10.2	Member function definitions for class Circle .	544
10.2	Definition of class Cylinder .	544
10.2	Member function definitions for class Cylinder .	545
10.2	Driver for point, circle, cylinder hierarchy.	546
Chapter 11	Stream Input/Output	
11.1	Portion of the stream I/O class hierarchy.	559
11.2	Portion of stream I/O class hierarchy with key file-processing classes.	560
11.3	Outputting a string using stream insertion.	561
11.4	Outputting a string using two stream insertions.	561
11.5	Using the endl stream manipulator.	562
11.6	Outputting expression values.	562
11.7	Concatenating the overloaded << operator.	563
11.8	Printing the address stored in a char * variable.	564
11.9	Calculating the sum of two integers input from the keyboard with cin and the stream-extraction operator.	565
11.10	Avoiding a precedence problem between the stream-insertion operator and the conditional operator.	565
11.11	Stream-extraction operator returning false on end-of-file.	566
11.12	Using member functions get , put , and eof .	567
11.13	Contrasting input of a string using cin with stream extraction and input with cin.get .	568

11.14	Character input with member function getline .	569
11.15	Unformatted I/O with the read , gcount and write member functions.	571
11.16	Using the hex , oct , dec and setbase stream manipulators.	572
11.17	Controlling precision of floating-point values.	573
11.18	Demonstrating the width member function.	574
11.19	Creating and testing user-defined, nonparameterized stream manipulators.	575
11.20	Format state flags.	576
11.21	Controlling the printing of trailing zeros and decimal points with float values.	577
11.22	Left-justification and right-justification.	578
11.23	Printing an integer with internal spacing and forcing the plus sign.	579
11.24	Using the fill member function and the setfill manipulator to change the padding character for fields larger than the values being printed.	580
11.25	Using the ios::showbase flag.	581
11.26	Displaying floating-point values in system default, scientific, and fixed formats.	582
11.27	Using the ios::uppercase flag.	583
11.28	Demonstrating the flags member function.	584
11.29	Testing error states.	586
11.30	User-defined stream-insertion and stream-extraction operators.	588
Chapter 12	Templates	
12.1	A template function.	606
12.2	Using template functions.	607
12.3	Definition of class template Stack .	609
12.3	Driver for class template Stack .	611
Chapter 13	Exception Handling	
13.1	A simple exception handling example with divide by zero.	628
Chapter 14	File Processing and String Stream I/O	
14.1	The data hierarchy.	652
14.2	C++'s view of a file of n bytes.	653
14.3	Portion of stream I/O class hierarchy.	653
14.4	Creating a sequential file.	654
14.5	File open modes.	655

14.6	End-of-file key combinations for various popular computer systems.	657
14.7	Reading and printing a sequential file.	659
14.8	Credit inquiry program.	661
14.9	Sample output of the credit inquiry program of Fig. 14.8.	663
14.10	C++'s view of a random access file.	664
14.11	Creating a random access file sequentially.	665
14.12	Writing data randomly to a random access file.	667
14.13	Sample execution of the program in Fig. 14.12.	668
14.14	Reading a random access file sequentially.	669
14.15	Bank account program.	671
14.16	Using a dynamically allocated <code>ostream</code> object.	676
14.17	Demonstrating an <code>ostream</code> object using a previously defined array.	677
14.18	Demonstrating input from an <code>istream</code> object.	678
Chapter 15	Data Structures	
15.1	Two self-referential class objects linked together.	694
15.2	A graphical representation of a list.	696
15.3	Manipulating a linked list.	697
15.4	Sample output for the program of Fig. 15.3.	703
15.5	The <code>insertAtFront</code> operation.	705
15.6	A graphical representation of the <code>insertAtBack</code> operation.	706
15.7	A graphical representation of the <code>removeFromFront</code> operation.	707
15.8	A graphical representation of the <code>removeFromBack</code> operation.	708
15.9	A simple stack program.	709
15.10	Sample output from the program of Fig. 15.9.	711
15.11	A simple stack program using composition.	712
15.12	Processing a queue.	713
15.13	Sample output from the program in Fig. 15.12.	715
15.14	A graphical representation of a binary tree.	716
15.15	A binary search tree.	717
15.16	Creating and traversing a binary tree.	718
15.17	Sample output from the program of Fig. 15.16.	721
15.18	A binary search tree.	722
15.19	A 15-node binary search tree.	727
15.20	Simple commands.	734
15.21	Simple program that determines the sum of two integers.	735
15.22	Simple program that finds the larger of two integers.	735

15.23	Calculate the squares of several integers.	736
15.24	Writing, compiling, and executing a Simple language program.	736
15.25	SML instructions produced after the compiler's first pass.	740
15.26	Symbol table for program of Fig. 15.25.	741
15.27	Unoptimized code from the program of Fig. 15.25.	744
15.28	Optimized code for the program of Fig. 15.25.	745
Chapter 16	Bits, Characters, Strings and Structures	
16.1	A possible storage alignment for a variable of type Example showing an undefined area in memory.	752
16.2	High-performance card shuffling and dealing simulation.	755
16.3	Output for the high-performance card shuffling and dealing simulation.	756
16.4	The bitwise operators.	757
16.5	Printing an unsigned integer in bits.	758
16.6	Results of combining two bits with the bitwise AND operator & .	759
16.7	Using the bitwise AND, bitwise inclusive OR, bitwise exclusive OR, and bitwise complement operators.	759
16.8	Output for the program of Fig. 16.7.	761
16.9	Results of combining two bits with the bitwise inclusive OR operator .	761
16.10	Results of combining two bits with the bitwise exclusive OR operator ^ .	762
16.11	Using the bitwise shift operators.	762
16.12	The bitwise assignment operators.	763
16.13	Operator precedence and associativity.	764
16.14	Using bit fields to store a deck of cards.	765
16.15	Output of the program in Fig. 16.14.	767
16.16	Summary of the character handling library functions.	768
16.17	Using isdigit , isalpha , isalnum , and isxdigit .	769
16.18	Using islower , isupper , tolower , and toupper .	770
16.19	Using isspace , iscntrl , ispunct , isprint , and isgraph .	772
16.20	Summary of the string conversion functions of the general utilities library.	773
16.21	Using atof .	774
16.22	Using atoi .	774
16.23	Using atol .	775

16.24	Using <code>strtod</code> .	776
16.25	Using <code>strtol</code> .	776
16.26	Using <code>strtoul</code> .	777
16.27	Search functions of the string handling library.	778
16.28	Using <code>strchr</code> .	779
16.29	Using <code>strcspn</code> .	779
16.30	Using <code>strpbrk</code> .	780
16.31	Using <code>strrchr</code> .	780
16.32	Using <code>strspn</code> .	781
16.33	Using <code>strstr</code> .	782
16.34	The memory functions of the string handling library.	782
16.35	Using <code>memcpy</code> .	783
16.36	Using <code>memmove</code> .	784
16.37	Using <code>memcmp</code> .	784
16.38	Using <code>memchr</code> .	785
16.39	Using <code>memset</code> .	785
16.40	Another string manipulation function of the string handling library.	786
16.41	Using <code>strerror</code> .	786
Chapter 17	The Preprocessor	
17.1	The predefined symbolic constants.	808
Chapter 18	Other Topics	
18.1	The type and the macros defined in header <code>stdarg.h</code> .	818
18.2	Using variable-length argument lists.	819
18.3	Using command-line arguments.	820
18.4	Using functions <code>exit</code> and <code>atexit</code> .	823
18.5	The signals defined in header <code>signal.h</code> .	825
18.6	Using signal handling.	826
18.7	Using <code>goto</code> .	828
18.8	Printing the value of a union in both member data types.	831
18.9	Using an anonymous union.	832
Appendix E	Number Systems	
E.1	Digits of the binary, octal, decimal, and hexadecimal number systems.	893
E.2	Comparison of the binary, octal, decimal, and hexadecimal number systems.	893
E.3	Positional values in the decimal number system.	893
E.4	Positional values in the binary number system.	894
E.5	Positional values in the octal number system.	894
E.6	Positional values in the hexadecimal number system.	895
E.7	Decimal, binary, octal, and hexadecimal equivalents.	895

E.8	Converting a binary number to decimal.	897
E.9	Converting an octal number to decimal.	897
E.10	Converting a hexadecimal number to decimal.	897