Contents

Chapters 22–27 and Appendices D–I are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

Preface xxi

1 Introduction to Computers, the Internet and the World Wide Web 1

1.1 Introduction 2
1.2 Computers: Hardware and Software 3
1.3 Computer Organization 4
1.4 Personal, Distributed and Client/Server Computing 5
1.5 The Internet and the World Wide Web 6
1.6 Web 2.0 6
1.7 Machine Languages, Assembly Languages and High-Level Languages 7
1.8 History of C and C++ 8
1.9 C++ Standard Library 9
1.10 Java 10
1.11 Fortran, COBOL, Pascal and Ada 10
1.12 BASIC, Visual Basic, Visual C++, C# and .NET 11
1.13 Key Software Trend: Object Technology 11
1.14 Typical C++ Development Environment 13
1.15 Notes About C++ and *C++ How to Program, Late Objects Version, 7/e* 15
1.16 Test-Driving a C++ Application 16
1.17 Software Technologies 22
1.18 Future of C++: Open Source Boost Libraries, TR1 and C++0x 23
1.19 Basic Object Technology Concepts 23
1.20 Wrap-Up 27
1.21 Web Resources 27

2 Introduction to C++ Programming 37

2.1 Introduction 38
2.2 First Program in C++: Printing a Line of Text 38
2.3 Modifying Our First C++ Program 42
2.4 Another C++ Program: Adding Integers 43
2.5 Memory Concepts 47
2.6 Arithmetic 48

2.7 Decision Making: Equality and Relational Operators 52
2.8 Wrap-Up 56

3 Control Statements: Part 1 66

3.1 Introduction 67
3.2 Algorithms 67
3.3 Pseudocode 68
3.4 Control Structures 69
3.5 `if` Selection Statement 72
3.6 `if...else` Double-Selection Statement 73
3.7 `while` Repetition Statement 78
3.8 Formulating Algorithms: Counter-Controlled Repetition 80
3.9 Formulating Algorithms: Sentinel-Controlled Repetition 84
3.10 Formulating Algorithms: Nested Control Statements 92
3.11 Assignment Operators 96
3.12 Increment and Decrement Operators 97
3.13 Wrap-Up 100

4 Control Statements: Part 2 115

4.1 Introduction 116
4.2 Essentials of Counter-Controlled Repetition 116
4.3 `for` Repetition Statement 118
4.4 Examples Using the `for` Statement 122
4.5 `do...while` Repetition Statement 126
4.6 `switch` Multiple-Selection Statement 128
4.7 `break` and `continue` Statements 135
4.8 Logical Operators 137
4.9 Confusing the Equality (==) and Assignment (=) Operators 141
4.10 Structured Programming Summary 142
4.11 Wrap-Up 147

5 Functions and an Introduction to Recursion 157

5.1 Introduction 158
5.2 Program Components in C++ 159
5.3 Math Library Functions 160
5.4 Function Definitions 161
5.5 Functions with Multiple Parameters 164
5.6 Function Prototypes and Argument Coercion 166
5.7 C++ Standard Library Header Files 169
5.8 Case Study: Random Number Generation 170
5.9 Case Study: Game of Chance; Introducing `enum` 175
5.10 Storage Classes 179
5.11 Scope Rules 182
5.12 Function Call Stack and Activation Records 185

5.13 Functions with Empty Parameter Lists 189
5.14 Inline Functions 189
5.15 References and Reference Parameters 191
5.16 Default Arguments 195
5.17 Unary Scope Resolution Operator 197
5.18 Function Overloading 198
5.19 Function Templates 201
5.20 Recursion 203
5.21 Example Using Recursion: Fibonacci Series 206
5.22 Recursion vs. Iteration 209
5.23 Wrap-Up 212

6 Arrays and Vectors 233

6.1 Introduction 234
6.2 Arrays 234
6.3 Declaring Arrays 236
6.4 Examples Using Arrays 237
- 6.4.1 Declaring an Array and Using a Loop to Initialize the Array's Elements 237
- 6.4.2 Initializing an Array in a Declaration with an Initializer List 238
- 6.4.3 Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations 239
- 6.4.4 Summing the Elements of an Array 241
- 6.4.5 Using Bar Charts to Display Array Data Graphically 242
- 6.4.6 Using the Elements of an Array as Counters 243
- 6.4.7 Using Arrays to Summarize Survey Results 244
- 6.4.8 Static Local Arrays and Automatic Local Arrays 247

6.5 Passing Arrays to Functions 249
6.6 Searching Arrays with Linear Search 253
6.7 Sorting Arrays with Insertion Sort 255
6.8 Multidimensional Arrays 257
6.9 Case Study: Processing Grades in a Two-Dimensional Array 260
6.10 Introduction to C++ Standard Library Class Template `vector` 265
6.11 Standard Library Class `string` 270
6.12 Wrap-Up 274

7 Pointers 291

7.1 Introduction 292
7.2 Pointer Variable Declarations and Initialization 292
7.3 Pointer Operators 293
7.4 Pass-by-Reference with Pointers 296
7.5 Using `const` with Pointers 300
7.6 Selection Sort Using Pass-by-Reference 304
7.7 `sizeof` Operator 308
7.8 Pointer Expressions and Pointer Arithmetic 311

7.9 Relationship Between Pointers and Arrays 313
7.10 Pointer-Based String Processing 316
7.11 Arrays of Pointers 319
7.12 Function Pointers 320
7.13 Wrap-Up 323

8 Sequential-Access Files 341

8.1 Introduction 342
8.2 Data Hierarchy 342
8.3 Files and Streams 344
8.4 Creating a Sequential File 345
8.5 Reading Data from a Sequential File 348
8.6 Updating Sequential Files 354
8.7 Wrap-Up 354

9 Classes 362

9.1 Introduction 363
9.2 Classes, Objects, Member Functions and Data Members 364
9.3 `Time` Class 365
9.4 Class Scope and Accessing Class Members 372
9.5 Placing a Class in a Separate File for Reusability 375
9.6 `Time` Class: Separating Interface from Implementation 375
9.7 Access Functions and Utility Functions 383
9.8 `Time` Class: Constructors with Default Arguments 386
9.9 Destructors 390
9.10 When Constructors and Destructors Are Called 391
9.11 `Time` Class: Using *Set* and *Get* Functions 394
9.12 `Time` Class: A Subtle Trap—Returning a Reference to a `private` Data Member 400
9.13 Default Memberwise Assignment 402
9.14 Wrap-Up 405

10 Classes: A Deeper Look 414

10.1 Introduction 415
10.2 `const` (Constant) Objects and `const` Member Functions 415
10.3 Composition: Objects as Members of Classes 424
10.4 `friend` Functions and `friend` Classes 430
10.5 Using the `this` Pointer 433
10.6 `static` Class Members 438
10.7 Data Abstraction and Information Hiding 443
10.8 Wrap-Up 445

11 Operator Overloading 451

11.1 Introduction 452

11.2 Fundamentals of Operator Overloading 453
11.3 Restrictions on Operator Overloading 454
11.4 Operator Functions as Class Members vs. Global Functions 455
11.5 Overloading Stream Insertion and Stream Extraction Operators 457
11.6 Overloading Unary Operators 460
11.7 Overloading Binary Operators 461
11.8 Dynamic Memory Management 461
11.9 Case Study: `Array` Class 463
11.10 Converting between Types 475
11.11 Building a `String` Class 476
11.12 Overloading ++ and -- 477
11.13 Case Study: A `Date` Class 479
11.14 `explicit` Constructors 483
11.15 Proxy Classes 487
11.16 Wrap-Up 490

12 Object-Oriented Programming: Inheritance 502

12.1 Introduction 503
12.2 Base Classes and Derived Classes 504
12.3 `protected` Members 507
12.4 Relationship between Base Classes and Derived Classes 507
 12.4.1 Creating and Using a `CommissionEmployee` Class 508
 12.4.2 Creating a `BasePlusCommissionEmployee` Class Without Using Inheritance 513
 12.4.3 Creating a `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy 518
 12.4.4 `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy Using `protected` Data 523
 12.4.5 `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy Using `private` Data 530
12.5 Constructors and Destructors in Derived Classes 537
12.6 `public`, `protected` and `private` Inheritance 545
12.7 Software Engineering with Inheritance 546
12.8 Wrap-Up 547

13 Object-Oriented Programming: Polymorphism 553

13.1 Introduction 554
13.2 Polymorphism Examples 555
13.3 Relationships Among Objects in an Inheritance Hierarchy 556
 13.3.1 Invoking Base-Class Functions from Derived-Class Objects 557
 13.3.2 Aiming Derived-Class Pointers at Base-Class Objects 564
 13.3.3 Derived-Class Member-Function Calls via Base-Class Pointers 565
 13.3.4 Virtual Functions 567
 13.3.5 Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers 573

13.4 Type Fields and switch Statements 574
13.5 Abstract Classes and Pure virtual Functions 574
13.6 Case Study: Payroll System Using Polymorphism 576
13.6.1 Creating Abstract Base Class Employee 578
13.6.2 Creating Concrete Derived Class SalariedEmployee 581
13.6.3 Creating Concrete Derived Class HourlyEmployee 583
13.6.4 Creating Concrete Derived Class CommissionEmployee 586
13.6.5 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee 588
13.6.6 Demonstrating Polymorphic Processing 589
13.7 (Optional) Polymorphism, Virtual Functions and Dynamic Binding "Under the Hood" 593
13.8 Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info 597
13.9 Virtual Destructors 601
13.10 Wrap-Up 601

14 Templates 607

14.1 Introduction 608
14.2 Function Templates 609
14.3 Overloading Function Templates 612
14.4 Class Templates 612
14.5 Nontype Parameters and Default Types for Class Templates 618
14.6 Notes on Templates and Inheritance 620
14.7 Notes on Templates and Friends 620
14.8 Notes on Templates and static Members 621
14.9 Wrap-Up 621

15 Stream Input/Output 626

15.1 Introduction 627
15.2 Streams 628
15.2.1 Classic Streams vs. Standard Streams 628
15.2.2 iostream Library Header Files 629
15.2.3 Stream Input/Output Classes and Objects 629
15.3 Stream Output 632
15.3.1 Output of char * Variables 632
15.3.2 Character Output Using Member Function put 632
15.4 Stream Input 633
15.4.1 get and getline Member Functions 633
15.4.2 istream Member Functions peek, putback and ignore 636
15.4.3 Type-Safe I/O 636
15.5 Unformatted I/O Using read, write and gcount 636
15.6 Introduction to Stream Manipulators 637
15.6.1 Integral Stream Base: dec, oct, hex and setbase 638

15.6.2 Floating-Point Precision (precision, setprecision) 639
15.6.3 Field Width (width, setw) 640
15.6.4 User-Defined Output Stream Manipulators 641
15.7 Stream Format States and Stream Manipulators 643
15.7.1 Trailing Zeros and Decimal Points (showpoint) 643
15.7.2 Justification (left, right and internal) 644
15.7.3 Padding (fill, setfill) 646
15.7.4 Integral Stream Base (dec, oct, hex, showbase) 647
15.7.5 Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed) 648
15.7.6 Uppercase/Lowercase Control (uppercase) 649
15.7.7 Specifying Boolean Format (boolalpha) 649
15.7.8 Setting and Resetting the Format State via Member Function flags 650
15.8 Stream Error States 652
15.9 Tying an Output Stream to an Input Stream 654
15.10 Wrap-Up 654

16 Exception Handling 664

16.1 Introduction 665
16.2 Exception-Handling Overview 666
16.3 Example: Handling an Attempt to Divide by Zero 666
16.4 When to Use Exception Handling 672
16.5 Rethrowing an Exception 673
16.6 Exception Specifications 675
16.7 Processing Unexpected Exceptions 675
16.8 Stack Unwinding 676
16.9 Constructors, Destructors and Exception Handling 678
16.10 Exceptions and Inheritance 678
16.11 Processing new Failures 679
16.12 Class auto_ptr and Dynamic Memory Allocation 682
16.13 Standard Library Exception Hierarchy 684
16.14 Other Error-Handling Techniques 686
16.15 Wrap-Up 686

17 Random-Access Files 693

17.1 Introduction 694
17.2 Random-Access Files 694
17.3 Creating a Random-Access File 695
17.4 Writing Data Randomly to a Random-Access File 699
17.5 Reading from a Random-Access File Sequentially 702
17.6 Case Study: A Transaction-Processing Program 704
17.7 Overview of Object Serialization 710
17.8 Wrap-Up 711

18 Class string and String Stream Processing 714
18.1 Introduction 715
18.2 string Assignment and Concatenation 716
18.3 Comparing strings 718
18.4 Substrings 721
18.5 Swapping strings 721
18.6 string Characteristics 722
18.7 Finding Substrings and Characters in a string 725
18.8 Replacing Characters in a string 727
18.9 Inserting Characters into a string 728
18.10 Conversion to C-Style Pointer-Based char * Strings 729
18.11 Iterators 731
18.12 String Stream Processing 732
18.13 Wrap-Up 735

19 Searching and Sorting 742
19.1 Introduction 743
19.2 Searching Algorithms 744
19.2.1 Efficiency of Linear Search 744
19.2.2 Binary Search 745
19.3 Sorting Algorithms 751
19.3.1 Efficiency of Selection Sort 751
19.3.2 Efficiency of Insertion Sort 751
19.3.3 Merge Sort (A Recursive Implementation) 752
19.4 Wrap-Up 759

20 Data Structures 764
20.1 Introduction 765
20.2 Self-Referential Classes 766
20.3 Dynamic Memory Allocation and Data Structures 767
20.4 Linked Lists 767
20.5 Stacks 782
20.6 Queues 787
20.7 Trees 790
20.8 Wrap-Up 799

21 Standard Template Library (STL) 810
21.1 Introduction to the Standard Template Library (STL) 811
21.1.1 Introduction to Containers 813
21.1.2 Introduction to Iterators 817
21.1.3 Introduction to Algorithms 822
21.2 Sequence Containers 824
21.2.1 vector Sequence Container 824

21.2.2 list Sequence Container 832
21.2.3 deque Sequence Container 836
21.3 Associative Containers 837
21.3.1 multiset Associative Container 838
21.3.2 set Associative Container 841
21.3.3 multimap Associative Container 842
21.3.4 map Associative Container 844
21.4 Container Adapters 845
21.4.1 stack Adapter 846
21.4.2 queue Adapter 848
21.4.3 priority_queue Adapter 849
21.5 Algorithms 851
21.5.1 fill, fill_n, generate and generate_n 851
21.5.2 equal, mismatch and lexicographical_compare 853
21.5.3 remove, remove_if, remove_copy and remove_copy_if 855
21.5.4 replace, replace_if, replace_copy and replace_copy_if 858
21.5.5 Mathematical Algorithms 860
21.5.6 Basic Searching and Sorting Algorithms 863
21.5.7 swap, iter_swap and swap_ranges 866
21.5.8 copy_backward, merge, unique and reverse 867
21.5.9 inplace_merge, unique_copy and reverse_copy 870
21.5.10 Set Operations 871
21.5.11 lower_bound, upper_bound and equal_range 874
21.5.12 Heapsort 876
21.5.13 min and max 879
21.5.14 STL Algorithms Not Covered in This Chapter 879
21.6 Class bitset 881
21.7 Function Objects 885
21.8 Wrap-Up 888
21.9 STL Web Resources 889

Chapters on the Web 899

A Operator Precedence and Associativity 900

B ASCII Character Set 902

C Fundamental Types 903

Appendices on the Web 905

Index 906

Chapters 22–27 and Appendices D–I are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

22 Bits, Characters, C Strings and `structs` 22-1

22.1 Introduction 22-2
22.2 Structure Definitions 22-2
22.3 Initializing Structures 22-5
22.4 Using Structures with Functions 22-5
22.5 `typedef` 22-5
22.6 Example: Card Shuffling and Dealing Simulation 22-6
22.7 Bitwise Operators 22-8
22.8 Bit Fields 22-18
22.9 Character-Handling Library 22-21
22.10 Pointer-Based String Manipulation Functions 22-27
22.11 Pointer-Based String-Conversion Functions 22-34
22.12 Search Functions of the Pointer-Based String-Handling Library 22-39
22.13 Memory Functions of the Pointer-Based String-Handling Library 22-44
22.14 Wrap-Up 22-48

23 Boost Libraries, Technical Report 1 and C++0x 23-1

23.1 Introduction 23-2
23.2 Deitel Online C++ and Related Resource Centers 23-2
23.3 Boost Libraries 23-2
23.4 Boost Libraries Overview 23-3
23.5 Regular Expressions with the `Boost.Regex` Library 23-6
    23.5.1 Regular Expression Example 23-6
    23.5.2 Validating User Input with Regular Expressions 23-9
    23.5.3 Replacing and Splitting Strings 23-12
23.6 Smart Pointers with `Boost.Smart_ptr` 23-14
    23.6.1 Reference Counted `shared_ptr` 23-14
    23.6.2 `weak_ptr`: `shared_ptr` Observer 23-19
23.7 Technical Report 1 23-24
23.8 C++0x 23-26
23.9 Core Language Changes 23-26
23.10 Wrap-Up 23-31

24 Other Topics 24-1

24.1 Introduction 24-2
24.2 `const_cast` Operator 24-2
24.3 `mutable` Class Members 24-4
24.4 `namespaces` 24-6
24.5 Operator Keywords 24-9
24.6 Pointers to Class Members (`.*` and `->*`) 24-11
24.7 Multiple Inheritance 24-13

24.8 Multiple Inheritance and `virtual` Base Classes 24-18
24.9 Wrap-Up 24-23

25 ATM Case Study, Part 1: Object-Oriented Design with the UML 25-1

25.1 Introduction 25-2
25.2 Examining the ATM Requirements Document 25-2
25.3 Identifying the Classes in the ATM Requirements Document 25-10
25.4 Identifying Class Attributes 25-17
25.5 Identifying Objects' States and Activities 25-21
25.6 Identifying Class Operations 25-25
25.7 Indicating Collaboration Among Objects 25-32
25.8 Wrap-Up 25-39

26 ATM Case Study, Part 2: Implementing an Object-Oriented Design 26-1

26.1 Introduction 26-2
26.2 Starting to Program the Classes of the ATM System 26-2
26.3 Incorporating Inheritance into the ATM System 26-9
26.4 ATM Case Study Implementation 26-16
26.4.1 Class `ATM` 26-16
26.4.2 Class `Screen` 26-24
26.4.3 Class `Keypad` 26-25
26.4.4 Class `CashDispenser` 26-26
26.4.5 Class `DepositSlot` 26-28
26.4.6 Class `Account` 26-29
26.4.7 Class `BankDatabase` 26-31
26.4.8 Class `Transaction` 26-35
26.4.9 Class `BalanceInquiry` 26-37
26.4.10 Class `Withdrawal` 26-39
26.4.11 Class `Deposit` 26-44
26.4.12 Test Program `ATMCaseStudy.cpp` 26-47
26.5 Wrap-Up 26-47

27 Game Programming with Ogre 27-1

27.1 Introduction 27-2
27.2 Installing Ogre, OgreAL and OpenAL 27-2
27.3 Basics of Game Programming 27-2
27.4 The Game of Pong: Code Walkthrough 27-5
27.4.1 Ogre Initialization 27-6
27.4.2 Creating a Scene 27-15
27.4.3 Adding to the Scene 27-16
27.4.4 Animation and Timers 27-28

27.4.5 User Input 27-29
27.4.6 Collision Detection 27-31
27.4.7 Sound 27-35
27.4.8 Resources 27-36
27.4.9 Pong Driver 27-37
27.5 Wrap-Up 27-38
27.6 Ogre Web Resources 27-38

D Number Systems D-1

D.1 Introduction D-2
D.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers D-5
D.3 Converting Octal and Hexadecimal Numbers to Binary Numbers D-6
D.4 Converting from Binary, Octal or Hexadecimal to Decimal D-6
D.5 Converting from Decimal to Binary, Octal or Hexadecimal D-7
D.6 Negative Binary Numbers: Two's Complement Notation D-9

E Preprocessor E-1

E.1 Introduction E-2
E.2 `#include` Preprocessor Directive E-2
E.3 `#define` Preprocessor Directive: Symbolic Constants E-3
E.4 `#define` Preprocessor Directive: Macros E-3
E.5 Conditional Compilation E-5
E.6 `#error` and `#pragma` Preprocessor Directives E-6
E.7 Operators # and ## E-7
E.8 Predefined Symbolic Constants E-7
E.9 Assertions E-8
E.10 Wrap-Up E-8

F C Legacy Code Topics F-1

F.1 Introduction F-2
F.2 Redirecting Input/Output on UNIX/Linux/Mac OS X and Windows Systems F-2
F.3 Variable-Length Argument Lists F-3
F.4 Using Command-Line Arguments F-5
F.5 Notes on Compiling Multiple-Source-File Programs F-7
F.6 Program Termination with `exit` and `atexit` F-9
F.7 Type Qualifier `volatile` F-10
F.8 Suffixes for Integer and Floating-Point Constants F-10
F.9 Signal Handling F-11
F.10 Dynamic Memory Allocation with `calloc` and `realloc` F-13
F.11 Unconditional Branch: `goto` F-14
F.12 Unions F-15
F.13 Linkage Specifications F-18
F.14 Wrap-Up F-19

G UML 2: Additional Diagram Types G-1
G.1 Introduction G-1
G.2 Additional Diagram Types G-1

H Using the Visual Studio Debugger H-1
H.1 Introduction H-2
H.2 Breakpoints and the **Continue** Command H-2
H.3 **Locals** and **Watch** Windows H-8
H.4 Controlling Execution Using the **Step Into**, **Step Over**, **Step Out** and **Continue** Commands H-11
H.5 **Autos** Window H-13
H.6 Wrap-Up H-14

I Using the GNU C++ Debugger I-1
I.1 Introduction I-2
I.2 Breakpoints and the `run`, `stop`, `continue` and `print` Commands I-2
I.3 `print` and `set` Commands I-8
I.4 Controlling Execution Using the `step`, `finish` and `next` Commands I-10
I.5 `watch` Command I-13
I.6 Wrap-Up I-15