Preface

This is a textbook for a one- or two-semester course in problem solving and program design. It is suitable for use by students with no programming background as well as those with a one-semester course, or the equivalent, in another programming language. Students’ backgrounds will determine the time required to cover the earlier chapters of the text and the extent of coverage possible for later chapters.

The earlier editions of this book represented the culmination of an eight-year effort, partially sponsored by the National Science Foundation, to define an introductory-level course that presented the rudimentary principles of software engineering and object-oriented programming along with an introduction to the C++ programming language. Our primary goal is to motivate and introduce sound principles of program design and abstraction in a first programming course. Early in the book we introduce topics such as program style, documentation, algorithm and data structuring, procedure- and data-oriented modularization, component reuse, and program verification. The focus throughout is on the problem solving/software design process, from problem analysis to program design and coding.

The textbook fits the objectives of the first course in programming, following an imperative first approach as described in the ACM/IEEE Computing Curricula 2001 (courses CS101 or CS111). Because it also introduces object-oriented programming and the C++ Standard Template Library (STL) as well as recursion and dynamic data structures, it may also be used as the textbook for the second course in the three-semester sequence, CS102. The textbook can also be used in an objects early approach to the first programming course as classes are introduced in Chapter 1 and used in Chapter 3. In that case, we recommend that you move Chapter 10, Class Definition and Use, forward. It could be studied anytime after Chapter 6.
New to the Sixth Edition

Students will benefit from a variety of new content, including:

- Graphics coverage implemented throughout the book. Many Computer Science faculty have recommended the use of graphics to help motivate the study of introductory programming and as a vehicle to help students understand how to use libraries and to call functions. We agree with this viewpoint and have included several optional sections with graphics examples in this edition. The new graphics sections include:
  
  Section 3.8: Introduction to Computer Graphics

  Section 5.10: Loops in Graphics Programs

  Section 7.6: Using Type char Data to Control a Graphics Program

  Section 9.12: Graphics Programs with Arrays

  To reduce the overhead required to introduce graphics, we decided to use WinBGlm (Windows BGI with mouse), which is a package based on the Turbo Pascal BGI (Borland Graphics Interface) library. WinBGlm was created to run on top of the Win32 library by Michael Main and his students at the University of Colorado. Several development platforms appropriate for CS 1 courses have incorporated WinBGlm. Quincy (developed by Al Stevens) is an open-source student-oriented C++ IDE that includes WinBGlm as well as more advanced libraries (http://www.codecutter.net/tools/quincy). A command-line platform based on the open-source GNU g++ compiler and the emacs program editor is distributed by the University of Colorado (http://www.codecutter.net/tools/winbgim). WinBGlm is also available for Bloodshed Software’s Dev-C++ and Microsoft’s Visual Studio C++.

- New Chapter 0 is designed to elucidate the opportunities and structures of the field of computing to introductory students. We hope this chapter encourages students to consider a major or minor in computing by illustrating a world in which they can envision themselves.

- Reorganization of coverage of multidimensional arrays and arrays of structs. These topics were moved from Chapter 11 (Data Abstraction and Object-Oriented Design) to Chapter 9 (Arrays and Structs) to keep all coverage of arrays and structs in one chapter and to enable students to complete their study of procedural programming topics before learning how to write classes.

- New Section 7.6 on iterative approximations including a case study on finding roots of an equation.

- New case study in Chapter 11 (An Address Book for an email Provider) that uses the C++ vector class as its underlying data structure.

- New end-of-section exercises and updated case studies.

- New end-of-chapter quick-check and review exercises and new programming project assignments including graphics programming assignments.
Balancing Object-Oriented and Procedural Approaches

Object-oriented concepts and classes are introduced early in the book, starting in Section 1.3. Sections 2.4 and 3.7 discuss the use of two system-defined classes, iostream and string, and we refer to the use of classes and objects throughout most of the text.

An issue of concern to faculty is the relative order of arrays, structs, and classes. As in the last edition, we introduce arrays and structs first (Chapter 9) and then introduce the definition and coding of classes (Chapter 10). Some faculty may prefer to reverse the order, and this is entirely possible. The chapter on classes uses arrays only in the implementation of class simpleString, which can be omitted or deferred until after arrays are covered.

We continue to emphasize the design of classes and data modeling in Chapter 11, which introduces template classes, an indexed-list class, the STL vector class, friend functions, and operator overloading. We also use template classes in Chapter 13, where we discuss dynamic data structures: lists, stacks, queues, and trees. We discuss the use of the STL container classes and iterators and also show students how to implement their own classes. An illustration of the C++ inheritance and virtual function mechanisms is provided in Appendix E in the publisher’s Web site. We’ve done our best to follow a balanced path between the strictly objects-first and totally procedure-focused programming metaphors. We agree with the objects-first concept, but not at the expense of the fundamentals of algorithm organization and design. Students in a first course can and should be taught the basic elements of procedural design. Our task is to do so within the context of an early focus on the importance of data modeling, reuse, and other fundamental principles of good software development.

Software Engineering and Object-Oriented Concepts

Many fundamental software engineering and object-oriented concepts are illustrated in the text: user-defined types, modeling problem-domain entities and their relationships, minimal interfaces, high-level cohesion, information hiding, separation of concerns, parameterized components, and inheritance. Abstraction is stressed from the start. Numerous complete case examples are provided throughout the text; these follow a standard software development method, from the specification and analysis of a problem to the first stage of design to the final coding.
Issues of program style are presented throughout in special displays. The concept of a program as a sequence of control structures is introduced in Chapter 3 and discussed in more detail in Chapters 4 (on selection structures) and 5 (on repetition structures). Our decision to introduce software engineering concepts in a first-year course is apparent in these early chapters. We've introduced functions and classes as early as possible at the introductory level—functions in Chapters 3 and 6 and the use and definition of classes in Chapters 3 and 10, respectively. We also provide several sections that discuss testing, debugging, and program verification.

Outline of Contents

Conceptually, the text may be partitioned into three sections. Chapters 1 through 6 provide introductory material on functions and top-down design, presenting detailed coverage of selection and repetition structures and program design strategies for using these structures. The connection between good problem-solving skills and effective software development is established early in the first three chapters. Included in the first two chapters are sections on problem solving and an introduction to a software development methodology based on a systematic approach to problem solving. The problem-solving approach outlined in these chapters is applied consistently to all other case studies in the text. Chapter 2 also contains an introduction to the basic elements of C++, including two sections in which we discuss abstraction, data modeling, and object-oriented programming. In Chapter 3, we continue the emphasis on basic problem-solving skills with a discussion of top-down design and divide and conquer. The reuse of program components is discussed, and additional detail is provided on the string class and its member functions.

Top-down procedural decomposition is further illustrated in Chapters 4 through 6. Decision structures are introduced in Chapter 4, and repetition structures are presented in Chapter 5. In Chapter 6, we revisit the C++ function, introducing functions with output arguments and providing a complete case study illustrating much of what has been learned to this point. An optional section on recursion is also included at the end of Chapter 6.

Chapters 7 through 9 cover simple data types, input and output, structured data types (arrays and structs), and classes. Chapter 7 contains a more detailed discussion of simple data types, including additional commentary on data abstraction as well as a description of the internal and external distinctions among the simple types. In Chapter 9, the structured types (arrays and structs) are first introduced. Simple searching and
sorting algorithms are discussed, and the use of structured types as function arguments is illustrated. Chapter 9 also covers multidimensional arrays and arrays of structs.

Chapter 8 provides an introduction to external file input/output. Although studying external files may seem premature at this point, we believe it is appropriate. Programs don’t exist in a vacuum; they manipulate data that often come from external sources, and they produce results that may subsequently be manipulated by other programs. It’s therefore important for students to gain a relatively early exposure to some fundamental concepts related to file input and output, as long as this exposure does not disrupt the presentation of other essential ideas. Of course, by the time students reach Chapter 8, they will already have been introduced to the basics of stream input and output, including a minimal use of formatting functions and input/output manipulators (Chapter 5).

For students with the equivalent of a one-semester programming course in another language, Chapters 1 through 9 can be covered fairly quickly, perhaps in as little as five or six weeks. For students with little or no background, this may take ten to twelve weeks.

Chapters 10 and 11 cover intermediate-level concepts that would normally be introduced at the end of CS1 or the beginning of CS2. Chapter 10 covers the definition and use of classes and objects. Chapter 11 focuses on data modeling. We begin with a discussion of template classes and develop our own indexed-list template class. Next, we introduce the Standard Template Library and provide a new case study which uses the C++ vector class to process an address book.

Chapters 12, 13, and 14 cover more advanced topics in some depth: recursion (Chapter 12), linked lists, stacks, queues, and trees (Chapter 13), and multiprocessing and threads (Chapter 14). Chapters 12 and 13 are normally covered in the second semester of the first-year sequence. Chapter 14 could also be included in this course or a later one on Operating Systems.

Coverage of Pointers

Pointers are introduced only where they really belong—in the discussion of dynamic data structures (Chapter 13). The pointer is one of the more dangerous, relatively unprotected aspects of the C++ language and need not be an essential part of an introductory text. Use of the new and delete operators and the allocation and deallocation of memory cells in the heap are discussed at the beginning of Chapter 13. We illustrate the manipulation of dynamic data structures such as simple linked lists, stacks and queues, and binary trees.
Pedagogical Features

Several pedagogical features also enhance the usefulness of the text as an instructional tool. These include the following:

- Consistent use of analysis and design aids such as data requirements tables and program structure charts
- End-of-section self-check and programming exercises (answers to the odd-numbered self-check exercises are available in the publisher’s Web site.)
- End-of-chapter quick-check exercises (answers are provided) and programming projects
- Numerous examples and case studies carried through from analysis and design to implementation
- Syntax displays containing the syntax and semantics of each new C++ feature introduced
- Program style and design guideline displays
- Detailed syntax and run-time error discussions at the end of each chapter
- Chapter reviews and review questions

Appendices

Separate appendices are available in the publisher’s Web site summarizing information about character sets, C++ reserved words, C++ operators, and function libraries (with descriptions and section numbers). There is also an appendix illustrating inheritance and virtual functions.

Supplemental Materials

The following supplements are available to all readers of this book at www.aw.com/cssupport:

- Source Code
- Answers to Odd-Numbered Questions
- Appendices

The following instructor supplements are only available to qualified instructors at Addison-Wesley’s Instructor Resource Center. Visit www.pearsonhighered.com/irc or send an e-mail to computing@aw.com for information about how to access them.

- PowerPoint Slides
- Instructor’s Manual with Solutions
- Test Bank
- Source Code
Acknowledgments

Many people participated in the development of this textbook. For this edition, we want to thank Michael Main for his assistance with WinBGIm and help with some of the graphics examples. We would also like to acknowledge the contributions of his students at the University of Colorado who adapted WinBGI to create WinBGIm (Grant Macklem, Gregory Schmelter, Alan Schmidt, and Ivan Stashak).

We also want to thank Charlotte Young of South Plains College for her help in creating Chapter 0, and Jeff Warsaw of Temple University and WaveRules, LLC, who contributed substantially to the chapter on multiprocessing in the fifth edition. Paul LaFollette and Paul Wolfgang of Temple University were primary contributors to the first edition. Temple graduate students Donna Chrupcala, Bruce Weiner, and Judith Wilson also contributed significantly to the development of the original textbook. Steve Vinoski provided detailed comments concerning the C++ material in many of the later chapters.

The principal reviewers and class testers have been enormously helpful in suggesting improvements and finding errors. For the first edition, these included Allen Alexander (Delaware Technical and Community College), Ruth Barton and Richard Reid (Michigan State University), Larry Cottrell (University of Central Florida), H. E. Dunsmore and Russell Quong (Purdue University), Donna Krabbe (College of Mount St. Joseph), Sally Kyvernitis (Neumann College), Xiaoping Jia (DePaul University), Xiannong Meng and Rick Zaccone (Bucknell), Jeff Buckwalter and Kim Summerhays (University of San Francisco), and Jo Ellen Perry (University of North Carolina). Valuable proofreading and editing assistance were provided by Sally Kyvernitis, Donna Skalski, and Frank Friedman’s daughters Dara and Shelley.

We are also very grateful to the principal reviewers of other editions of this book for their hard work and timely responses. They include William E. Bulley (Merit Network, Inc.), Greg Comeau (Comeau Computing), Bruce Gilland (University of Colorado at Boulder), William I. Grosky (Wayne State University), Bina Ramamurthy (SUNY at Buffalo), and W. Brent Seales (University of Kentucky).

Frank Friedman is particularly indebted to several members of the staff at the Software Engineering Institute (Pittsburgh), particularly Mary Shaw, Norm Gibbs (Guilford College), and Gary Ford, for their support during the year in which the seeds that lead to this book were sown.

As always, it has been a pleasure working with the people of Addison-Wesley throughout this endeavor. Editor-in-Chief Michael Hirsch, along
Preface

with Editorial Assistant Stephanie Sellinger, provided guidance and encouragement throughout all phases of manuscript revision. Heather McNally and Wanda Rockwell supervised the production of the book, while Margaret Waples and Kathryn Ferranti developed the marketing campaign.

F. L. F.
E. B. K.
Brief Contents

chapter zero  Computer Science as a Career Path  1
chapter one  Introduction to Computers, Problem Solving, and Programming  15
chapter two  Overview of C++  53
chapter three  Top-Down Design with Functions and Classes  117
chapter four  Selection Structures: if and switch Statements  197
chapter five  Repetition and Loop Statements  259
chapter six  Modular Programming  333
chapter seven  Simple Data Types  387
chapter eight  Streams and Files  443
chapter nine  Data Structures: Arrays and Structs  487
chapter ten  User-Defined Classes  565
chapter eleven  Data Abstraction and Object-Oriented Design  627
chapter twelve  Recursion  691
chapter thirteen  Pointers and Dynamic Data Structures  729
chapter fourteen  Multiprocessing Using Processes and Threads  795
Index  851

The following are available online:

Appendix A  ASCII Character Set
Appendix B  Reserved Words and Special Characters
Appendix C  Selected C++ Library Facilities
Appendix D  Operators
Appendix E  A Brief Introduction to Inheritance and Polymorphism

Answers to Odd-Numbered Questions
Contents

chapter zero

Computer Science as a Career Path  1

chapter one

Introduction to Computers, Problem Solving, and Programming  15

1.1 Overview of Computers  16
   Early Computers  16
   Categories of Computers  17
   Sharing Computer Resources  18

1.2 Computer Hardware  18
   Memory  20
   Main Memory  21
   Secondary Memory  22
   Central Processing Unit  24
   Input/Output Devices  24
   Computer Networks  25
   The World Wide Web  27

1.3 Computer Software  28
   Operating System  28
   Application Software  30
   Programming Languages  31
   Object-Oriented Programming  32

1.4 Processing a High-Level Language Program  35
   Executing a Program  37

1.5 The Software Development Method  38
   Caution: Failure Is Part of the Process  41

1.6 Applying the Software Development Method  41
   CASE STUDY: Converting Miles to Kilometers  42
chapter two

Overview of C++ 53

2.1 C++ Language Elements 54
   Comments 54
   Compiler Directive #include 55
   Namespace std 56
   Function main 56
   Declaration Statements 57
   Executable Statements 58

2.2 Reserved Words and Identifiers 60
   Reserved Words 60
   Identifiers 60
   Uppercase and Lowercase Letters 61

2.3 Data Types and Declarations 63
   Data Types 63
   string Class 66
   Purpose of Data Types 67
   Declarations 67
   Constant Declarations 68

2.4 Executable Statements 71
   Programs in Memory 71
   Assignment Statements 71
   Input/Output Operations 74
   Input Statements 74
   Program Output 76
   The return Statement 78

2.5 General Form of a C++ Program 80
   Comments in Programs 81

2.6 Arithmetic Expressions 83
   Operators / and % 84
## Contents

- Data Type of a Mixed-Type Expression 86
- Mixed-Type Assignment Statement 86
- Expressions with Multiple Operators 87
- Writing Mathematical Formulas in C++ 91
  
  **CASE STUDY: Supermarket Coin Processor** 93

2.7 Interactive Mode, Batch Mode, and Data Files 98
  
  Input Redirection 98
  
  Output Redirection 100

2.8 Common Programming Errors 101
  
  Syntax Errors 102
  
  Run-Time Errors 103
  
  Undetected Errors 104
  
  Logic Errors 105

Chapter Review 106
  
  Quick-Check Exercises 108
  
  Review Questions 109

Programming Projects 110
  
  Answers to Quick-Check Exercises 113

Josée Lajoie 115

---

### chapter three

**Top-Down Design with Functions and Classes** 117

3.1 Building Programs from Existing Information 118
  
  **CASE STUDY: Finding the Area and Circumference of a Circle** 119
  
  **CASE STUDY: Computing the Weight of a Batch of Flat Washers** 121

3.2 Library Functions 126
  
  C++ Library Functions 128
  
  A Look Ahead 130

3.3 Top-Down Design and Structure Charts 132
  
  **CASE STUDY: Drawing Simple Figures** 132

3.4 Functions without Arguments 134
  
  Function Prototypes 136
  
  Function Definitions 136
  
  Placement of Functions in a Program 138
  
  Order of Execution of Functions 138
  
  Advantages of Using Function Subprograms 140

3.5 Functions with Input Arguments 144
  
  **void Functions with Input Arguments** 146
Contents

Functions with Input Arguments and a Single Result 146
Functions with Multiple Arguments 151
Argument/Parameter List Correspondence 152
The Function Data Area 153
Testing Functions Using Drivers 153

3.6 Scope of Names 154

3.7 Extending C++ through Classes: Using Class string 157
The string Class 157
Declaring string Objects 159
Reading and Displaying string Objects 159
String Assignment and Concatenation 160
Operator Overloading 160
Dot Notation: Calling Functions length and at 160
Member Functions for Word-Processing
Operations 161
Assigning a Substring to a string Object 162

3.8 Introduction to Computer Graphics (Optional) 164
Composition of a Window 164
Some Common Graphics Functions 165
Background Color and Foreground Color 167
Drawing Rectangles 168
Drawing Circles, Ellipses, and Arcs 169
Pie Slices and Filled Ellipses 176
Adding Text to Drawings 178

3.9 Common Programming Errors 181
Separately Testing Function Subprograms 183

Chapter Review 184
Quick-Check Exercises 185
Review Questions 186
Programming Projects 187
Graphics Projects 191
Answers to Quick-Check Exercises 192

Mark Hall 194

chapter four

Selection Structures: if and switch Statements 197

4.1 Control Structures 198
4.2 Logical Expressions 198
Logical Expressions Using Relational and Equality Operators 199
Contents

Logical Expressions Using Logical Operators 200
Operator Precedence  202
Writing Conditions in C++  204
Comparing Characters and Strings  206
Boolean Assignment  207
Writing bool Values  208
Using Integers to Represent Logical Values  208

4.3 Introduction to the if Control Statement 209
  if Statement with Two Alternatives 210
  if Statement with Dependent Statement  211
  if Statement Conditions with Characters
  and Strings  211
  Format of the if Statement  213

4.4 if Statements with Compound Alternatives 215
  Tracing an if Statement 216

4.5 Decision Steps in Algorithms 219
  case study: Payroll Problem with Functions  219
  A Reminder about Identifier Scope  227
  Adding Data Flow Information to Structure Charts  227
  Commentary—The Software Development Method  228

4.6 Checking the Correctness of an Algorithm 229

4.7 Nested if Statements and Multiple-Alternative Decisions 231
  Comparison of Nested if Statements and a Sequence
  of if Statements  232
  Writing a Nested if as a Multiple-Alternative
  Decision 232
  Order of Conditions 234
  Short-Circuit Evaluation of Logical Expressions 237

4.8 The switch Control Statement 239
  Proper Use of break 242
  Comparison of Nested if Statements and the switch
  Statement  242
  Using a switch Statement to Select Alternative
  Functions  242

4.9 Common Programming Errors 244
  Chapter Review 245
  Quick-Check Exercises  248
  Review Questions  250

Programming Projects 251
  Answers to Quick-Check Exercises 256
chapter five  Repetition and Loop Statements  259
5.1 Counting Loops and the while Statement  260
   The while Statement  261
   Syntax of the while Statement  262
5.2 Accumulating a Sum or Product in a Loop  265
   Multiplying a List of Numbers  268
   Compound Assignment Operators  269
5.3 The for Statement  271
   Increment and Decrement Operators  273
   Increments and Decrements Other than One  275
   Displaying a Table of Values  276
5.4 Conditional Loops  279
   A Loop with a Decreasing Loop Control
   Variable  280
   CASE STUDY: Monitoring Oil Supply  280
   Zero Iteration Loops  282
   More General Conditional Loops  284
5.5 Loop Design and Loop Patterns  286
   Sentinel-Controlled Loops  287
   Calculating an Average  290
   Flag-Controlled Loops  291
5.6 The do-while Statement  293
5.7 Review of while, for, and do-while Loops  299
5.8 Nested Loops  301
5.9 Debugging and Testing Programs  306
   Using a Debugger  306
   Debugging without a Debugger  307
   Off-by-One Errors  308
   Testing  309
5.10 Loops in Graphics Programs (Optional)  309
   Animation  312
5.11 Common Programming Errors  316
   Chapter Review  319
   Quick-Check Exercises  321
   Review Questions  322
   Programming Projects  324
   Graphics Projects  330
   Answers to Quick-Check Exercises  330

Mike Weisert  331
chapter six

Modular Programming 333

6.1 Value and Reference Parameters 334
   Call-by-Value and Call-by-Reference Parameters 336
   void Functions Can Return Results 338
   When to Use a Reference or a Value Parameter 338
   Comparison of Value and Reference Parameters 339
   Protection Afforded by Value Parameters 339
   Argument/Parameter List Correspondence Revisited 340

6.2 Functions with Output and Input Parameters 344

6.3 Stepwise Design with Function 351
   CASE STUDY: General Sum and Average Problem 352
   Multiple Declarations of Identifiers in a Program 359

6.4 Using Objects with Functions 361

6.5 Debugging and Testing a Program System 363
   Top-Down Testing and Stubs 363
   Bottom-Up Testing and Drivers 364
   Debugging Tips for Program Systems 365
   Identifier Scope and Watch Window Variables 366
   Black-Box versus White-Box Testing 366

6.6 Recursive Functions (Optional) 368

6.7 Common Programming Errors 371
   Chapter Review 372
   Quick-Check Exercises 374
   Review Questions 375
   Programming Projects 376
   Answers to Quick-Check Exercises 384

Robert Sebesta 385

chapter seven

Simple Data Types 387

7.1 Constants Revisited 388
   The #define Compiler Directive 388

7.2 Internal Representations of Numeric Data Types 389
   Fixed-Point and Floating-Point Data Types 389
   Integer Types 390
   Floating-Point Types 391
   Types of Numeric Literals 391
chapter ten

User-Defined Classes 565

10.1 Class Definition and Use 566
The counter Class 566
Class Definition for the counter Class 567
Compiler Directives in File counter.h 570
Using the counter Class 571
Compiler Directives in File CounterTest.cpp 572

10.2 Class Implementation 573
Constructors 575
Accessor and Modifier Functions 576
Compiler Directives in File counter.cpp 577
chapter twelve

Recursion 691

12.1 The Nature of Recursion 692
   Properties of Recursive Problems and Solutions 693

12.2 Tracing Recursive Functions 695
   Tracing a Recursive Function 695
   Displaying Characters in Reverse Order 696
   Stack for Function Calls 699
   Implementation of Argument Stacks in C++ 701

12.3 Recursive Mathematical Functions 702

12.4 Recursive Functions with Array Arguments 708
   CASE STUDY: Binary Search 710

12.5 Problem Solving with Recursion 715
   CASE STUDY: The Towers of Hanoi 715
   Comparison of Iteration and Recursive Functions 721
chapter thirteen  Pointers and Dynamic Data Structures  729

13.1 Pointers and the new Operator  730
   Accessing Data with Pointers  731
   Pointer Operations  732
   Pointers with Arrays  732
   Pointers to Structs  733

13.2 Manipulating the Heap  737
   Effect of the new Operator on the Heap  737
   Returning Cells to the Heap  738

13.3 Linked Lists and the list Class  739
   Declaring Nodes  739
   Connecting Nodes  740
   Inserting a Node in a List  741
   Insertion at the Head of a List  742
   Insertion at the End of a List  743
   Deleting a Node  744
   Traversing a List  744
   Circular Lists and Two-Way Lists (Optional)  746
   The list Class  747

13.4 The Stack Abstract Data Type  750
   The C++ stack Class  751
   Implementing a stack Template Class  755
   Implementing the Stack Operators  755
   Testing the Stack ADT  758

13.5 The Queue Abstract Data Type  760
   The C++ queue Class  761
   Implementing a Queue ADT  761

13.6 Binary Trees  767
   Binary Search Tree  768
   Searching a Binary Search Tree  768
   Building a Binary Search Tree  770
   Displaying a Binary Search Tree  771
13.7 Binary Search Tree Abstract Data Type 773
   Design of Binary Tree Class 773
   Implementation of Binary Tree Class 776
13.8 Efficiency of a Binary Search Tree 781
13.9 Common Programming Errors 783
   Syntax Errors 783
   Run-Time Errors 783
Chapter Review 784
   Quick-Check Exercises 785
   Review Questions 787
   Programming Projects 789
   Answers to Quick-Check Exercises 793

Chapter fourteen  Multiprocessing Using Processes and Threads 795
14.1 Multitasking 796
   Linear versus Parallel Programming 796
   Multitasking as Time Sharing 797
   Preemptive Multitasking 798
   Time Slices and Parallelism 798
   Concurrent Programming 800
14.2 Processes 801
   Creating a Process 802
   Waiting for a Process 804
   Executing Another Program from a Process 806
14.3 Interprocess Communications and Pipes 808
   Pipes 808
   Using Pipes 809
   Interprocess Communications Using Standard Input 812
   Illustrating Communication between a Parent and a Child Process 813
14.4 Threads 817
   Creating a Thread 818
   Thread Synchronization 821
   Mutual Exclusion Locking 822
   Deadlocks 825
   CASE STUDY: Threads Illustrated 829
14.5 Common Programming Errors 843
   Chapter Review 843
   Review of C++ Constructs 844
The following are available online:

Appendix A  ASCII Character Set
Appendix B  Reserved Words and Special Characters
Appendix C  Selected C++ Library Facilities
Appendix D  Operators
Appendix E  A Brief Introduction to Inheritance and Polymorphism
  E.1 Subclassing for Specialization
  E.2 Subclassing for Specification
    CASE STUDY: Areas and Perimeters of Different Figures
  E.3 Commentary
  E.4 Polymorphism

Answers to Odd-Numbered Questions