Contents

Preface ix

Chapter Dependency Chart xi

PART ONE
Problem-Solving Techniques 1

1 Principles of Programming and Software Engineering 3

1.1 Software Engineering and Object-Oriented Design 4
An Examination of Problem Solving 4
Aspects of an Object-Oriented Solution 4
Abstraction and Information Hiding 5
Principles of Object-Oriented Programming 7
Object-Oriented Analysis and Design 8
Applying the UML to OOA/D 9
The Software Life Cycle 19
Iterative and Evolutionary Development 19
Rational Unified Process Development Phases 20
What About the Waterfall Method of Development? 23

1.2 Achieving a Better Solution 24
Evaluation of Designs and Solutions 24
Operation Contracts 27
Verification 29
What Is a Good Solution? 32

1.3 Key Issues in Programming 34
Modularity 35
Style 36
Modifiability 45
Ease of Use 47
Fail-Safe Programming 48
Debugging 53
Testing 55

2 Recursion: The Mirrors 65

2.1 Recursive Solutions 66
A Recursive Valued Function: The Factorial of *n* 69
A Recursive `void` Function: Writing a String Backward 76

2.2 Counting Things 85
Multiplying Rabbits (The Fibonacci Sequence) 85
Organizing a Parade 87
Mr. Spock's Dilemma (Choosing *k* Out of *n* Things) 90

2.3 Searching an Array 93
Finding the Largest Item in an Array 93
Binary Search 94
Finding the k^{th} Smallest Item of an Array 98

2.4 Organizing Data 102
The Towers of Hanoi 102

2.5 Recursion and Efficiency 106

3 Data Abstraction: The Walls 121

3.1 Abstract Data Types 122

3.2 Specifying ADTs 127
The ADT List 128
The ADT Sorted List 133
Designing an ADT 134
Axioms (Optional) 139

3.3 Implementing ADTs 141
C++ Classes 143
C++ Namespaces 152
An Array-Based Implementation of the ADT List 154
C++ Exceptions 160
An Implementation of the ADT List Using Exceptions 162

4 Linked Lists 171

4.1 Preliminaries 172
Pointers 173
Dynamic Allocation of Arrays 180
Pointer-Based Linked Lists 182

4.2 Programming with Linked Lists 184
Displaying the Contents of a Linked List 184
Deleting a Specified Node from a Linked List 186
Inserting a Node into a Specified Position of a Linked List 189
A Pointer-Based Implementation of the ADT List 194
Comparing Array-Based and Pointer-Based Implementations 202
Saving and Restoring a Linked List by Using a File 205
Passing a Linked List to a Method 208
Processing Linked Lists Recursively 209
Objects as Linked List Data 214

4.3 Variations of the Linked List **215**
Circular Linked Lists 216
Dummy Head Nodes 217
Doubly Linked Lists 218
4.4 Application: Maintaining an Inventory **221**
4.5 The C++ Standard Template Library **227**
Containers 228
Iterators 229
The Standard Template Library Class *list* 230

5 Recursion as a Problem-Solving Technique 247

5.1 Backtracking **248**
The Eight Queens Problem 248
Implementing Eight Queens Using the STL Class *vector* 250
5.2 Defining Languages **256**
The Basics of Grammars 256
Two Simple Languages 258
Algebraic Expressions 260
5.3 The Relationship Between Recursion and Mathematical Induction **270**
The Correctness of the Recursive Factorial Function 270
The Cost of Towers of Hanoi 271

PART TWO
Problem Solving with Abstract Data Types 283

6 Stacks 285

6.1 The Abstract Data Type Stack **286**
Developing an ADT During the Design of a Solution 286
6.2 Simple Applications of the ADT Stack **292**
Checking for Balanced Braces 292
Recognizing Strings in a Language 294
6.3 Implementations of the ADT Stack **296**
An Array-Based Implementation of the ADT Stack 297
A Pointer-Based Implementation of the ADT Stack 301
An Implementation That Uses the ADT List 305
Comparing Implementations 308
The Standard Template Library Class *stack* 309
6.4 Application: Algebraic Expressions **311**
Evaluating Postfix Expressions 311
Converting Infix Expressions to Equivalent Postfix Expressions 313
6.5 Application: A Search Problem **316**
A Nonrecursive Solution That Uses a Stack 317
A Recursive Solution 327

6.6 The Relationship Between Stacks and Recursion **329**

7 Queues 343

7.1 The Abstract Data Type Queue **344**

7.2 Simple Applications of the ADT Queue **346**
Reading a String of Characters 346
Recognizing Palindromes 347

7.3 Implementations of the ADT Queue **348**
A Pointer-Based Implementation 349
An Array-Based Implementation 354
An Implementation That Uses the ADT List 361
The Standard Template Library Class *queue* 364
Comparing Implementations 367

7.4 A Summary of Position-Oriented ADTs **368**

7.5 Application: Simulation **369**

8 Advanced C++ Topics 387

8.1 Inheritance Revisited **388**
Public, Private, and Protected Inheritance 395
Is-a, Has-a, and *As-a* Relationships 395

8.2 Virtual Methods and Late Binding **398**
Abstract Base Classes 404

8.3 Friends **408**

8.4 The ADTs List and Sorted List Revisited **411**
Implementations of the ADT Sorted List That Use the ADT List 413

8.5 Class Templates **419**

8.6 Overloaded Operators **426**

8.7 Iterators **431**
Implementing the ADT List Using Iterators 433

9 Algorithm Efficiency and Sorting 445

9.1 Measuring the Efficiency of Algorithms **446**
The Execution Time of Algorithms 447
Algorithm Growth Rates 448
Order-of-Magnitude Analysis and Big O Notation 450
Keeping Your Perspective 454
The Efficiency of Searching Algorithms 456

9.2 Sorting Algorithms and Their Efficiency **458**
Selection Sort 459
Bubble Sort 462
Insertion Sort 464
Mergesort 466
Quicksort 472
Radix Sort 484
A Comparison of Sorting Algorithms 486
The Standard Template LibrarySorting Algorithms 487

10 Trees 499

10.1 Terminology **500**

10.2 The ADT Binary Tree **508**

Traversals of a Binary Tree 512

Possible Representations of a Binary Tree 515

A Pointer-Based Implementation of the ADT Binary Tree 519

10.3 The ADT Binary Search Tree **536**

Algorithms for the ADT Binary Search Tree Operations 539

A Pointer-Based Implementation of the ADT Binary Search Tree 555

The Efficiency of Binary Search Tree Operations 564

Treesort 568

Saving a Binary Search Tree in a File 569

The STL Search Algorithms 572

10.4 General Trees **575**

11 Tables and Priority Queues 589

11.1 The ADT Table **590**

Selecting an Implementation 595

A Sorted Array-Based Implementation of the ADT Table 602

A Binary Search Tree Implementation of the ADT Table 607

11.2 The ADT Priority Queue: A Variation of the ADT Table **610**

Heaps 614

A Heap Implementation of the ADT Priority Queue 623

Heapsort 626

11.3 Tables and Priority Queues in the STL **630**

The STL Associative Containers 630

The STL *priority_queue* Class and Heap Algorithms 638

12 Advanced Implementations of Tables 649

12.1 Balanced Search Trees **650**

2-3 Trees 651

2-3-4 Trees 670

Red-Black Trees 678

AVL Trees 681

12.2 Hashing **686**

Hash Functions 690

Resolving Collisions 693

The Efficiency of Hashing 701

What Constitutes a Good Hash Function? 704

Table Traversal: An Inefficient Operation Under Hashing 706

Implementing a *HashMap* Class Using the STL 707

12.3 Data with Multiple Organizations **710**

13 Graphs 721

13.1 Terminology **722**

13.2 Graphs as ADTs **725**

Implementing Graphs 726

Implementing a Graph Class Using the STL 729

13.3 Graph Traversals **732**

Depth-First Search 733

Breadth-First Search 736

Implementing a BFS Class Using the STL 737

13.4 Applications of Graphs **740**

Topological Sorting 740

Spanning Trees 743

Minimum Spanning Trees 747

Shortest Paths 749

Circuits 754

Some Difficult Problems 756

14 Processing Data in External Storage 765

14.1 A Look at External Storage **766**

14.2 Sorting Data in an External File **769**

14.3 External Tables **776**

Indexing an External File 779

External Hashing 783

B-Trees 787

Traversals 797

Multiple Indexing 799

A Review of C++ Fundamentals 807

B ASCII Character Codes 880

C C++ Header Files and Standard Functions 881

D Mathematical Induction 887

E Standard Template Library 893

F C++ Documentation Systems 905

Glossary 909

Answers to Self-Test Exercises 935

Index 953