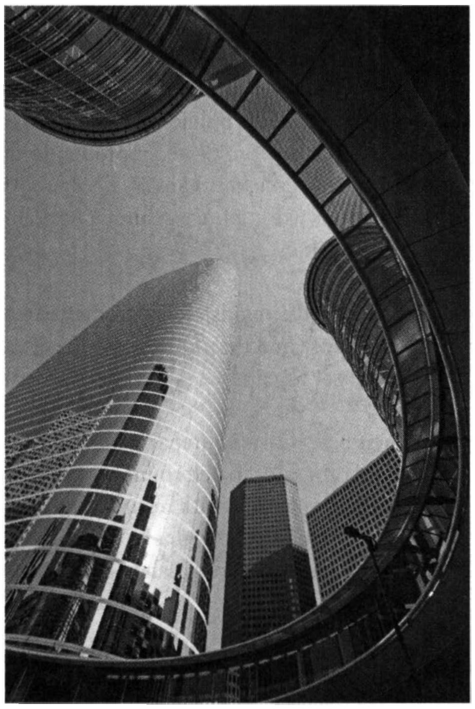
Data Abstraction & Problem Solving with C++

Walls And Mirrors

Seventh Edition



Frank M. Carrano
University of Rhode Island

Timothy M. Henry
New England Institute of Technology

PEARSON

Boston Columbus Indianapolis Hoboken New York San Francisco
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial Director, ECS: Marcia J. Horton
Executive Editor: Tracy Johnson (Dunkelberger)
Editorial Assistant: Kristy Alaura
Program Manager: Carole Snyder
Project Manager: Robert Engelhardt
Media Team Lead: Steve Wright
R&P Manager: Rachel Youdelman
R&P Senior Project Manager: William Opaluch
Senior Operations Specialist: Maura Zaldivar-Garcia
Inventory Manager: Meredith Maresca
Marketing Manager: Demetrius Hall
Product Marketing Manager: Bram Van Kempen
Marketing Assistant: Jon Bryant
Cover Designer: Marta Samsel
Cover Art: © Jeremy Woodhouse/Ocean/Corbis
Full-Service Project Management: John Orr, Cenveo Publisher Services / Nesbitt Graphics, Inc.

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text.

© 2017 Pearson Education, Inc.
Hoboken, New Jersey 07030

All rights reserved. Printed in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson PLC, Permissions Department, 330 Hudson St, New York, NY 10013.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Pearson Education Ltd., London
Pearson Education Australia Ply. Ltd., Sydney
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., Hong Kong
Pearson Education Canada, Inc., Toronto
Pearson Education de Mexico, S.A. de C.V.
Pearson Education–Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., Hoboken, New Jersey

Library of Congress Cataloging-in-Publication Data on file

10 9 8 7 6 5 4 3 2 1

PEARSON www.pearsonhighered.com

ISBN-10: 0-13-446397-8
ISBN 13: 978-0-13-446397-1

Contents

Chapter 1 Data Abstraction: The Walls **1**

1.1 Object-Oriented Concepts 2

1.1.1 Object-Oriented Analysis and Design 2

1.1.2 Aspects of an Object-Oriented Solution 3

1.2 Achieving a Better Solution 4

1.2.1 Cohesion 4

1.2.2 Coupling 5

1.3 Specifications 6

1.3.1 Operation Contracts 6

1.3.2 Unusual Conditions 8

1.3.3 Abstraction 9

1.3.4 Information Hiding 10

1.3.5 Minimal and Complete Interfaces 12

1.4 Abstract Data Types 12

1.4.1 Designing an ADT 15

1.4.2 ADTs That Suggest Other ADTs 17

1.5 The ADT Bag 18

1.5.1 Identifying Behaviors 18

1.5.2 Specifying Data and Operations 20

1.5.3 An Interface Template for the ADT 23

1.5.4 Using the ADT Bag 25

C++ Interlude 1 C++ Classes **31**

C1.1 A Problem to Solve 32

C1.1.1 Private Data Fields 33

C1.1.2 Constructors and Destructors 34

C1.1.3 Methods 34

C1.1.4 Preventing Compiler Errors 35

C1.2 Implementing a Solution 36

C1.2.1 Initializers 37

C1.3 Templates 38

C1.4 Inheritance 40

C1.4.1 Base Classes and Derived Classes 41

C1.4.2 Overriding Base-Class Methods 43

C1.5 Virtual Methods and Abstract Classes 44

C1.5.1 Virtual Methods 45

C1.5.2 Abstract Classes 45

Chapter 2 Recursion: The Mirrors **47**

2.1 Recursive Solutions 48

2.2 Recursion That Returns a Value 50

2.2.1 A Recursive Valued Function: The Factorial of *n* 50

2.2.2 The Box Trace 54

2.3 Recursion That Performs an Action 57

2.3.1 A Recursive Void Function: Writing a String Backward 57

2.4 Recursion with Arrays 67

2.4.1 Writing an Array's Entries in Backward Order 67

2.4.2 Searching a Sorted Array: The Binary Search 68

2.4.3 Finding the Largest Value in a Sorted or Unsorted Array 72
2.4.4 Finding the k^{th} Smallest Value of an Unsorted Array 74
2.5 Organizing Data 76
2.5.1 The Towers of Hanoi 76
2.6 More Examples 79
2.6.1 The Fibonacci Sequence (Multiplying Rabbits) 80
2.6.2 Organizing a Parade 81
2.6.3 Choosing k Out of n Things 84
2.7 Recursion and Efficiency 86

Chapter 3 Array-Based Implementations 95
3.1 The Approach 96
3.1.1 Core Methods 97
3.1.2 Using Fixed-Size Arrays 98
3.2 An Array-Based Implementation of the ADT Bag 98
3.2.1 The Header File 99
3.2.2 Defining the Core Methods 101
3.2.3 Testing the Core Methods 103
3.2.4 Implementing More Methods 105
3.2.5 Methods That Remove Entries 107
3.2.6 Testing 110
3.3 Using Recursion in the Implementation 112
3.3.1 The Method `getIndexOf` 112
3.3.2 The Method `getFrequencyOf` 113

C++ Interlude 2 Pointers, Polymorphism, and Memory Allocation 117
C2.1 Memory Allocation for Variables and Early Binding of Methods 118
C2.2 A Problem to Solve 118
C2.3 Pointers and the Program's Free Store 120
C2.3.1 Deallocating Memory 122
C2.3.2 Avoiding Memory Leaks 123
C2.3.3 Avoiding Dangling Pointers 126
C2.4 Virtual Methods and Polymorphism 128
C2.5 Dynamic Allocation of Arrays 130
C2.5.1 A Resizable Array-Based Bag 131

Chapter 4 Link-Based Implementations 133
4.1 Preliminaries 134
4.1.1 The Class `Node` 136
4.2 A Link-Based Implementation of the ADT Bag 138
4.2.1 The Header File 138
4.2.2 Defining the Core Methods 139
4.2.3 Implementing More Methods 143
4.3 Using Recursion in Link-Based Implementations 148
4.3.1 Recursive Definitions of Methods in `LinkedBag` 148
4.4 Testing Multiple ADT Implementations 150
4.5 Comparing Array-Based and Link-Based Implementations 153

Chapter 5 Recursion as a Problem-Solving Technique 159
5.1 Defining Languages 160
5.1.1 The Basics of Grammars 160
5.1.2 Two Simple Languages 162

5.2 Algebraic Expressions 164
5.2.1 Kinds of Algebraic Expressions 164
5.2.2 Prefix Expressions 166
5.2.3 Postfix Expressions 170
5.2.4 Fully Parenthesized Expressions 172
5.3 Backtracking 172
5.3.1 Searching for an Airline Route 172
5.3.2 The Eight Queens Problem 178
5.4 The Relationship Between Recursion and Mathematical Induction 181
5.4.1 The Correctness of the Recursive Factorial Function 181
5.4.2 The Cost of Towers of Hanoi 182

Chapter 6 Stacks 191
6.1 The Abstract Data Type Stack 192
6.1.1 Developing an ADT During the Design of a Solution 192
6.1.2 Specifications for the ADT Stack 194
6.2 Simple Uses of a Stack 199
6.2.1 Checking for Balanced Braces 199
6.2.2 Recognizing Strings in a Language 201
6.3 Using Stacks with Algebraic Expressions 203
6.3.1 Evaluating Postfix Expressions 203
6.3.2 Converting Infix Expressions to Equivalent Postfix Expressions 204
6.4 Using a Stack to Search a Flight Map 208
6.5 The Relationship Between Stacks and Recursion 214

C++ Interlude 3 Exceptions 225
C3.1 Background 226
C3.1.1 A Problem to Solve 226
C3.2 Assertions 227
C3.3 Throwing Exceptions 228
C3.4 Handling Exceptions 231
C3.4.1 Multiple catch Blocks 234
C3.4.2 Uncaught Exceptions 235
C3.4.3 Where to Handle Exceptions 237
C3.5 Programmer-Defined Exception Classes 238

Chapter 7 Implementations of the ADT Stack 241
7.1 An Array-Based Implementation 242
7.2 A Link-Based implementation 245
7.3 Implementations That Use Exceptions 249

Chapter 8 Lists 253
8.1 Specifying the ADT List 254
8.2 Using the List Operations 259
8.3 An Interface Template for the ADT List 261

Chapter 9 List Implementations 265
9.1 An Array-Based Implementation of the ADT List 266
9.1.1 The Header File 266
9.1.2 The Implementation File 268
9.2 A Link-Based Implementation of the ADT List 272
9.2.1 The Header File 273

9.2.2 The Implementation File 274
9.2.3 Using Recursion in `LinkedList` Methods 282
9.3 Comparing Implementations 286

C++ Interlude 4 Safe Memory Management Using Smart Pointers 291
C4.1 Raw Pointers 292
C4.2 Smart Pointers 292
C4.2.1 Using Shared Pointers 293
C4.2.2 Revised `Node` and `LinkedList` Classes 296
C4.2.3 Using Unique Pointers 301
C4.2.4 Using Weak Pointers 302
C4.3 Other Smart Pointer Features 304
C4.3.1 A Method Common to All Smart Pointers 304
C4.3.2 A Method Common to All Shared and Unique Pointers 305
C4.3.3 Methods Exclusive to Shared Pointers 305
C4.3.4 A Method Exclusive to Unique Pointers 306
C4.3.5 Unique Pointers with Arrays 306

Chapter 10 Algorithm Efficiency 307
10.1 What Is a Good Solution? 308
10.2 Measuring the Efficiency of Algorithms 309
10.2.1 The Execution Time of Algorithms 310
10.2.2 Algorithm Growth Rates 311
10.2.3 Analysis and Big O Notation 313
10.2.4 Keeping Your Perspective 317
10.2.5 The Efficiency of Searching Algorithms 318

Chapter 11 Sorting Algorithms and Their Efficiency 325
11.1 Basic Sorting Algorithms 326
11.1.1 The Selection Sort 326
11.1.2 The Bubble Sort 329
11.1.3 The Insertion Sort 331
11.2 Faster Sorting Algorithms 333
11.2.1 The Merge Sort 333
11.2.2 The Quick Sort 338
11.2.3 The Radix Sort 347
11.3 A Comparison of Sorting Algorithms 349

C++ Interlude 5 Class Relationships and Reuse 353
C5.1 Inheritance Revisited 353
C5.1.1 Public, Private, and Protected Sections of a Class 360
C5.1.2 Public, Private, and Protected Inheritance 361
C5.1.3 *Is-a* and *As-a* Relationships 362
C5.2 Containment: *Has-a* Relationships 363
C5.3 Abstract Base Classes Revisited 365

Chapter 12 Sorted Lists and Their Implementations 367
12.1 Specifying the ADT Sorted List 368
12.1.1 An Interface Template for the ADT Sorted List 370
12.1.2 Using the Sorted List Operations 371
12.2 A Link-Based Implementation 373
12.2.1 The Header File 373

12.2.2 The Implementation File 374
12.2.3 The Efficiency of the Link-Based Implementation 377
12.3 Implementations That Use the ADT List 377
12.3.1 Containment 378
12.3.2 Public Inheritance 382
12.3.3 Private Inheritance 386

Chapter 13 Queues and Priority Queues 393
13.1 The ADT Queue 394
13.2 Simple Applications of the ADT Queue 397
13.2.1 Reading a String of Characters 397
13.2.2 Recognizing Palindromes 397
13.3 The ADT Priority Queue 399
13.3.1 Tracking Your Assignments 400
13.4 Application: Simulation 401
13.5 Position-Oriented and Value-Oriented ADTs 409

Chapter 14 Queue and Priority Queue Implementations 417
14.1 Implementations of the ADT Queue 418
14.1.1 An Implementation That Uses the ADT List 418
14.1.2 A Link-Based Implementation 420
14.1.3 An Array-Based Implementation 425
14.1.4 Comparing Implementations 430
14.2 An Implementation of the ADT Priority Queue 431

C++ Interlude 6 Overloaded Operators and Friend Access 435
C6.1 Overloaded Operators 435
C6.1.1 Overloading = for Assignment 437
C6.1.2 Overloading + for Concatenation 439
C6.2 Friend Access and Overloading << 441

Chapter 15 Trees 445
15.1 Terminology 446
15.1.1 Kinds of Trees 448
15.1.2 The Height of Trees 450
15.1.3 Full, Complete, and Balanced Binary Trees 451
15.1.4 The Maximum and Minimum Heights of a Binary Tree 453
15.2 The ADT Binary Tree 456
15.2.1 Traversals of a Binary Tree 456
15.2.2 Binary Tree Operations 459
15.2.3 An Interface Template for the ADT Binary Tree 461
15.3 The ADT Binary Search Tree 463
15.3.1 Binary Search Tree Operations 464
15.3.2 Searching a Binary Search Tree 466
15.3.3 Creating a Binary Search Tree 467
15.3.4 Traversals of a Binary Search Tree 468
15.3.5 The Efficiency of Binary Search Tree Operations 469

Chapter 16 Tree Implementations 475
16.1 The Nodes in a Binary Tree 476
16.1.1 An Array-Based Representation 476
16.1.2 A Link-Based Representation 478

16.2 A Link-Based Implementation of the ADT Binary Tree 479
16.2.1 The Header File 479
16.2.2 The Implementation 482
16.3 A Link-Based Implementation of the ADT Binary Search Tree 491
16.3.1 Algorithms for the ADT Binary Search Tree Operations 491
16.3.2 The Class `BinarySearchTree` 502
16.4 Saving a Binary Search Tree in a File 504
16.5 Tree Sort 507
16.6 General Trees 508

Chapter 17 Heaps 515
17.1 The ADT Heap 516
17.2 An Array-Based Implementation of a Heap 519
17.2.1 Algorithms for the Array-Based Heap Operations 520
17.2.2 The Implementation 524
17.3 A Heap Implementation of the ADT Priority Queue 528
17.4 Heap Sort 530

C++ Interlude 7 Iterators 537
C7.1 Iterators 537
C7.1.1 Common Iterator Operations 538
C7.1.2 Using Iterator Operations 540
C7.1.3 Implementing an Iterator 541
C7.2 Advanced Iterator Functionality 543

Chapter 18 Dictionaries and Their Implementations 547
18.1 The ADT Dictionary 548
18.1.1 An Interface for the ADT Dictionary 552
18.2 Possible Implementations 554
18.2.1 A Sorted Array-Based Implementation of the ADT Dictionary 557
18.2.2 A Binary Search Tree Implementation of the ADT Dictionary 559
18.3 Selecting an Implementation 562
18.3.1 Three Scenarios 562
18.4 Hashing as a Dictionary Implementation 568
18.4.1 Hash Functions 570
18.4.2 Resolving Collisions Using Open Addressing 573
18.4.3 Resolving Collisions By Restructuring the Hash Table 577
18.4.4 The Efficiency of Hashing 578
18.4.5 What Constitutes a Good Hash Function? 581
18.4.6 Dictionary Traversal: An Inefficient Operation Under Hashing 583
18.4.7 Using Hashing and Separate Chaining to Implement the ADT Dictionary 583

Chapter 19 Balanced Search Trees 591
19.1 Balanced Search Trees 592
19.2 AVL Trees 593
19.3 2-3 Trees 596
19.3.1 Traversing a 2-3 Tree 599
19.3.2 Searching a 2-3 Tree 600
19.3.3 Adding Data to a 2-3 Tree 602
19.3.4 Removing Data from a 2-3 Tree 608

19.4 2-3-4 Trees 614
19.4.1 Searching and Traversing a 2-3-4 Tree 616
19.4.2 Adding Data to a 2-3-4 Tree 616
19.4.3 Removing Data from a 2-3-4 Tree 619
19.5 Red-Black Trees 621
19.5.1 Searching and Traversing a Red-Black Tree 623
19.5.2 Adding to and Removing from a Red-Black Tree 623

Chapter 20 Graphs 629
20.1 Terminology 630
20.2 Graphs as ADTs 633
20.2.1 Implementing Graphs 635
20.3 Graph Traversals 638
20.3.1 Depth-First Search 638
20.3.2 Breadth-First Search 641
20.4 Applications of Graphs 642
20.4.1 Topological Sorting 642
20.4.2 Spanning Trees 645
20.4.3 Minimum Spanning Trees 650
20.4.4 Shortest Paths 651
20.4.5 Circuits 656
20.4.6 Some Difficult Problems 657

Chapter 21 Processing Data in External Storage 665
21.1 A Look at External Storage 666
21.2 Working with External Data 668
21.2.1 Sorting Data in an External File 668
21.2.2 Basic Data Management Operations 674
21.2.3 Indexing an External File 676
21.2.4 External Hashing 680
21.2.5 B-Trees 682
21.2.6 Traversals 690
21.2.7 Multiple Indexing 692

C++ Interlude 8 The Standard Template Library 697
C8.1 STL Container 697
C8.1.1 STL Container Adapters 699
C8.1.2 Sequence Containers 700
C8.1.3 Associative Containers 704
C8.2 STL Algorithms 708

Appendix A Review of C++ Fundamentals 713

Appendix B Important Themes in Programming 751

Appendix C The Unified Modeling Language 773

Appendix D The Software Life Cycle 781

Appendix E Mathematical Induction 787

Appendix F Algorithm Verification 791

Appendix G C++ File Fundamentals 795

Contents

Appendix H	C++ Header Files and Standard Functions	805
Appendix I	C++ Documentation Systems	809
Appendix J	ASCII Character Codes	811
Appendix K	C++ for Java Programmers	813
Appendix L	C++ for Python Programmers	821
Index		828
Glossary		Available online
Answers to Checkpoint Questions		Available online