

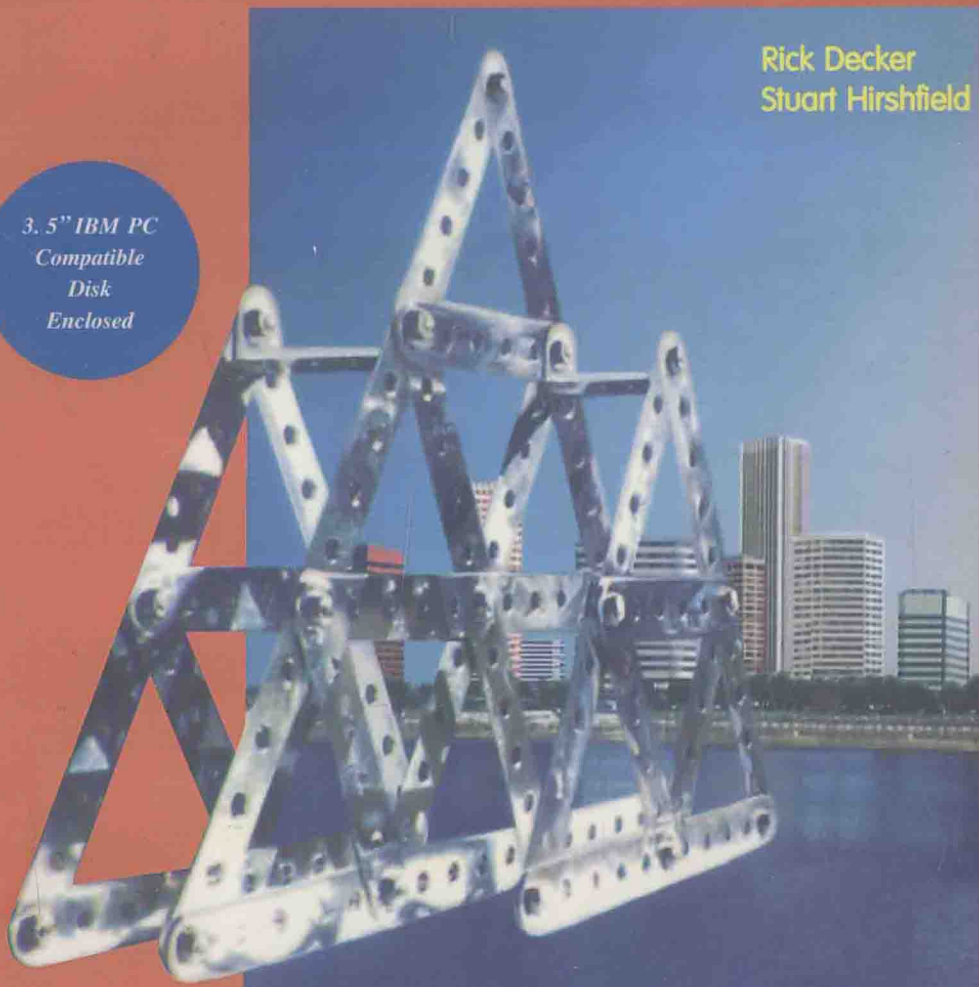
# Working Classes

DATA STRUCTURES AND ALGORITHMS USING C++

工作类型: 数据结构与算法的 C++ 实现

Rick Decker  
Stuart Hirshfield

3.5" IBM PC  
Compatible  
Disk  
Enclosed



I T P

世界图书出版公司



# WORKING CLASSES

*Data Structures and Algorithms Using C++*

**RICK DECKER**  
**STUART HIRSHFIELD**

*Hamilton College*



PWS Publishing Company

I(T)P International Thomson Publishing Company

书 名: Working Classes : Data Structure and Algorithms  
作 者: R.Decker, S.Hirshfield  
中译名: 工作类型: 数据结构与算法的C++实现  
出版者: 世界图书出版公司北京公司  
印刷者: 北京中西印刷厂  
发 行: 世界图书出版公司北京公司 (北京朝阳门内大街 137 号 100010)  
开 本: 大 32 开 850 × 1168 印 张: 16.125  
版 次: 1998 年 8 月第 1 版 1998 年 8 月第 1 次印刷  
书 号: 7-5062-3819-5/TP·34  
版权登记: 图字 01-98-0369  
定 价: 88.00 元

世界图书出版公司北京公司已获得 International Thomson Publishing 授权在中国境内独家重印发行。



PWS PUBLISHING COMPANY  
20 Park Plaza, Boston, Massachusetts 02116-4324

Copyright © 1997 by International Thomson Publishing Inc.

ALL RIGHTS RESERVED. No part of this book may be reproduced or transmitted photocopying, recording or any information storage and retrieval system, without permission, in writing, from the publisher.

**ITP**™ International Thomson Publishing  
The trademark ITP is used under license.

For more information contact:

**PWS Publishing Co.**  
20 Park Plaza  
Boston, MA 02116

**International Thomson Publishing Japan**  
Hirakawacho Kyowa Building, 31  
2-2-1 Hirakawacho  
Chiyoda-ku, Tokyo 102  
Japan

**Thomson Nelson Australia**  
102 Dodds Street  
South Melbourne, 3205  
Victoria, Australia

**International Thomson Publishing Europe**  
Berkshire House 168-173  
High Holborn  
London WC1V 7AA  
England

**International Thomson Editores**  
Campos Eliseos 385, Piso 7  
Col. Polanco  
11560 Mexico D.F., Mexico

**Nelson Canada**  
1120 Birchmount Road  
Scarborough, Ontario  
Canada M1K 5G4

**International Thomson Publishing Asia**  
221 Henderson Road  
#05-10 Henderson Building  
Singapore 0315

**International Thomson Publishing GmbH**  
Königswinterer Strasse 418  
53227 Bonn, Germany

**Library of Congress Cataloging-in-Publication Data**

Decker, Rick.

Working classes: data structures and algorithms using C++ / Rick Decker,

Stuart Hirshfield

p. cm.

Includes index.

ISBN 0-534-94566-X

1. C++ (Computer program language) 2. Data structures (Computer science)

I. Hirshfield, Stuart. II. Title.

QA76.73.C153D44 1996

005.7'3—dc20

94-43941

CIP



This book is printed  
on recycled, acid-free  
paper.

*Sponsoring Editor:* Michael J. Sugarman  
*Developmental Editor:* Mary Thomas  
*Production Editor:* Abigail M. Heim  
*Marketing Manager:* Nathan Wilbur  
*Manufacturing Coordinator:* Lisa Flanagan  
*Editorial Assistant:* Benjamin Steinberg

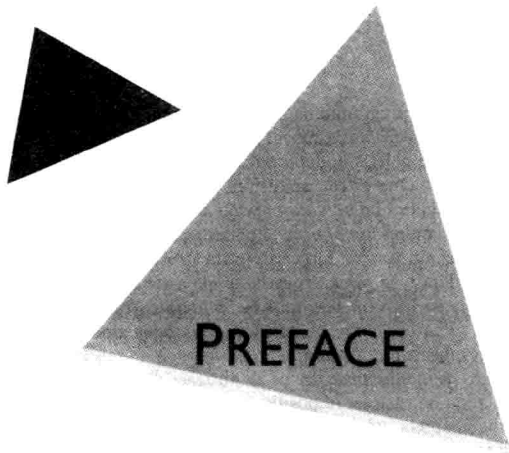
*Interior Designer:* Catherine Hawkes Design  
*Cover Designer:* Julia Gecha  
*Cover Artist:* Angela Perkins  
*Typesetter and Interior Illustrator:* Electric Ink, Ltd.  
*Cover Printer:* New England Book Components  
*Text Printer and Binder:* Quebecor Printing/Martinsburg

*Cover Image:* The SteelTec™ product ©1993 by Remco  
Toys, Inc. All rights reserved. Used with permission.

Printed and bound in the United States of America.

97 98 99—9 8 7 6 5 4

*For Natty, Adam, Ben, and Shauna*



## PREFACE

**T**he overwhelming majority of authors have very little to say. If we suppose, rather charitably, that in a typical book of fifteen chapters there are only eight passages worthy of quotation, then simple mathematics will convince us that in short order there will be no original quotations left for chapter headings. The implication is obvious...

Armand Blaquiere  
How To Write

Over the years, we've had a number of students who have said, in one form or another, "I want to be a computer scientist because I really like programming and am very good at it." Of course computer scientists, both novices and seasoned veterans, are often called upon to write programs, but to equate computer science with programming is to confuse the product with the process. Being an excellent draftsman who can faithfully represent a scene on paper is no guarantee that your works will eventually hang in the Metropolitan Museum. It's a step in the right direction, but an artist must also have an intimate familiarity with the more general principles of composition, perspective, color and so on.

In essence, programming is little more than the efficient management of a particular kind of large intellectual process, and the guidelines for good programming are nothing but the application of common-sense principles that apply to any complex creative task. It goes without saying, though, that before you can think efficiently you have to have something to think *about*, which for our purposes means that in order to write good programs, you must have an idea about how information may be represented in a program.

Computer science is a young discipline, but has developed enough over the past few decades to gain a consensus about what should constitute the core data structures. In this book, we have tried to capture this core by providing what might be called the "classic" data structures—the most commonly applied methods for representing information in a computer program—along with the algorithms for manipulating this information. In terms of things to think about for programming, this book offers a collection of tools that should be part of the working knowledge of any programmer.

This book is not about programming, however. Computer science is a science, and as such mainly seeks a theoretical framework that can be used to describe the behavior of the objects under study, which in our case are computers and their programs. One of the objectives that have determined the form of the book is to provide a broad view of what a data structure really is. In our approach, data structures are not just a collection of ad hoc type declarations and function definitions, but rather any data structure is a particular instance of an abstract data type, which consists of (1) a set of positions and a set of elements associated with the positions; (2) a logical structure defined on the positions; and (3) a collection of structure-preserving operations on the positions and the elements they “contain.”

We have chosen to define the structure of an abstract data type by specifying a structural relation on each set of positions. Doing so provides a natural progression of the chapters, where each new abstract data type is introduced by removing some of the structural restrictions from a prior type. Thus we begin with lists, whose structure is defined by a linear order, and progress to trees by removing the requirement that each position have a unique successor, then to directed graphs by removing the requirement of a unique predecessor, and finally to sets, where there is no structure at all on the positions. Throughout this process, we see that each new abstract data type still can be described by the threefold view of a collection of positions with a structural relation and a collection of structure-preserving operations.

### **Some History**

After using Pascal in this course for five years, it was clear to us that, for all its strengths as a teaching language, Pascal is not the most felicitous choice as a vehicle for a course in data structures. An abstract data type is nothing more than a collection of data and operations on that data, and that, of course, is the definition of a class. When preparing to write the book you have before you, we considered several object-oriented languages and finally settled on C++, largely because of its popularity. We'd be the first to admit that C++ has its warts and blemishes, but in our opinion it is the appropriate choice at present.

### **The Audience**

Though we did not set out to tailor this book to any preexisting curriculum, it turned out that it covers essentially all of CS2 and part of CS7, as described in the ACM Curriculum '78, and a subset of the union of CS2 and CO2, set forth in Norman Gibbs and Alan Tucker's 1985 Model Curriculum for a Liberal Arts Degree in Computer Science. The material contained here should be covered early in any computer science curriculum, and we have written this book for an audience of first and second year students in computer science

who are familiar with C or (preferably) C++. For those readers whose background is Pascal, we provide a Pascal-C++ “dictionary” in Appendix A. A course in discrete mathematics is desirable as a pre- or corequisite for this material, but the relevant mathematical background is summarized in Appendices B and C for those who need it.

## The Contents

Our intent has been to write a book that could be used as the basis for a semester-length course in data structures or advanced programming. Realizing that the subject matter of this book comes at an early stage in the education of a computer scientist, we included a number of mentions, necessarily brief, of some of the topics awaiting the student down the road. Most of the canonical sorting and searching algorithms are covered, along with mentions of computational complexity, compiler design, unsolvable problems, NP-completeness, and fundamental paradigms for algorithms. We believe that one can never have enough exercises—this book has 359, by actual count, and each chapter concludes with an optional Explorations section, where we treat interesting topics that extend the material of the chapter.

Chapter 1 covers some of the necessary preliminaries, such as program design, the definition of an abstract data type, and assertions and program verification. We begin by specifying an array as an abstract data type, and conclude with the *Number* ADT that represents integers of arbitrary size. Chapter 2 describes the *List* ADT and continues the preliminary material of Chapter 1 by discussing parametrized classes and functions, big-O notation, and timing of algorithms. The chapter concludes with a discussion of memory management. In the Explorations section, we discuss sorted lists and searching, along with self-organizing lists.

In Chapters 3 and 4 we continue the investigation of linear data structures. Chapter 3 covers strings and introduces the Boyer-Moore string search algorithm. Chapter 4 covers the remaining standard linear structures, stacks and queues, motivating these by applications to manipulate postfix expressions. The Explorations cover stack-based maze traversal and a simple operating system simulation. Since a considerable number of queue applications involve simulation, Appendix C (Random Numbers and Simulation) may be useful at this point.

Chapter 5 provides a segue into nonlinear structures by introducing recursion and recursively defined data structures. Timing estimates for recursive algorithms are covered in depth, along with an introduction to LISP. We deal with Quicksort in the Explorations. Appendix B, which covers logarithms and exponentials, induction, and elementary combinations, is helpful supplementary material at this stage.

Chapters 6 and 7 cover trees. Chapter 6 provides the necessary background on binary trees and their implementations, traversal algorithms, and treesort; and the Explorations discuss threaded trees, minimal-length codes,



and tries. Chapter 7, which can be omitted if necessary, covers two extensions of binary search trees, namely AVL trees and B-trees.

Chapter 8 covers graphs and digraphs, along with a representative sample of graph algorithms for traversal, spanning trees, minimal-cost paths, minimal spanning trees, and an introduction to complexity theory through the Traveling Salesperson Problem. In the Explorations, we discuss topological sorting and applications of powers of the adjacency matrix.

Chapter 9, on sets, describes bit vector, list implementations of sets, dictionaries, and associations, and provides a comprehensive introduction to hashing. The chapter concludes with *PriorityQueue* ADT and heapsort. In the Explorations, we continue our discussion of hashing and introduce the *DisjointSet* ADT.

In Chapter 10 we consider the problem of regenerating text from a large sample and trace the development of programs to solve this problem, using a real computer/compiler system to show how practical time and space constraints arise from choices of data structure.

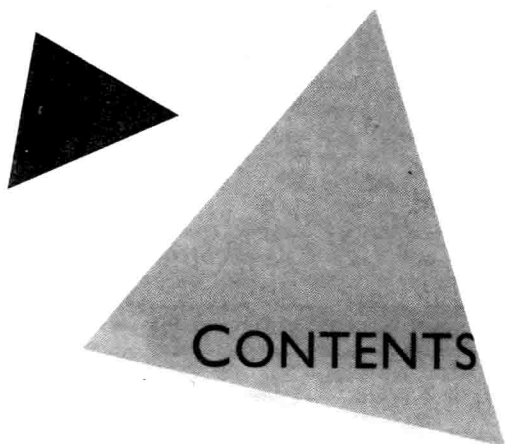
### Supplementary Material

In addition to the data disk (IBM PC compatible) included with this book, an *Instructor's Manual* is available from the publisher. A Macintosh version of the data disk is also available from the publisher.

### Acknowledgments

A lot of people deserve praise for seeing this book through to completion. Thanks go to Billy Lim, *Illinois State University*, Barbara Boucher Owens, *St. Edward's University*, and Daniel Ling, *Okanagan University College*, for their thoughtful reviews; and to our students and colleagues for suggesting countless changes in earlier versions. Special kudos go to the folks at PWS Publishing, especially Mike Sugarman and Ben Steinberg (the Batman and Robin of publishing), Abby Heim (who held her nervous breakdown at bay throughout an insanely busy production process that included working on two of our books simultaneously), J. P. Lenney (for picking out great wines and picking up the tab), and Nathan Wilbur (for just being Nathan). Writing and producing a book is a task that rates up there on the discomfort scale with cholera, except that writing takes longer. It can never be called pleasurable, but the friendship and warmth of the PWS crew at least has made it bearable.

Rick Decker  
Stuart Hirshfield



PART  
ONE

## INTRODUCTION

### PRELIMINARIES 3

- 1.1 **ADTs: ABSTRACTION AND ENCAPSULATION 4**
  - Abstraction 5
  - Reuse and Encapsulation 7
  - ADTs, OOP, and Things to Come 7
- 1.2 **ADT: *INTEGERARRAY* 8**
- 1.3 **IMPLEMENTATION 13**
  - Defining Integer Arrays 13
- 1.4 **COMPUTER SCIENCE INTERLUDE: ASSERTIONS AND VERIFICATION 18**
  - Assertions 18
  - Verification 19
- 1.5 **APPLICATION: MULTIPRECISION ARITHMETIC 23**
  - Declaring the *Number* Class 25
  - Defining the *Number* Class 27
- 1.6 **SUMMARY 36**

- 1.7 EXERCISES 36
- 1.8 EXPLORATIONS 44
  - Representation of Integers 44
  - Bit Vectors 45

PART  
TWO**LINEAR STRUCTURES****2**

- LISTS 49
  - 2.1 ADT: *LIST* 50
    - Parametrized Classes 53
  - 2.2 IMPLEMENTATIONS 55
    - Arrays 55
    - Linked Lists 63
  - 2.3 COMPARING IMPLEMENTATIONS 75
    - Space 75
    - Time 76
    - Comprehensibility 77
    - Trade-Offs 78
  - 2.4 COMPUTER SCIENCE INTERLUDE: MEASURES OF EFFICIENCY 78
    - Algorithms 79
    - Big-O 81
    - Order Arithmetic 83
    - Timing Functions 85
  - 2.5 APPLICATION: MEMORY MANAGEMENT 89
    - Allocation 92
    - Deallocation 94
    - Compaction 98
  - 2.6 SUMMARY 100
  - 2.7 EXERCISES 100
  - 2.8 EXPLORATIONS 111
    - Sorted Lists 111
    - Self-Organizing Lists 115

**3****STRINGS 117**

- 3.1 ADT: *STRING* 117**
  - (S)trings, (s)trings, and Arrays 118
  - Lexicographic Order 121
  - Declaring Strings 122
- 3.2 IMPLEMENTATION 124**
  - Efficiency 130
- 3.3 APPLICATION: STRING MATCHING 132**
- 3.4 SUMMARY 139**
- 3.5 EXERCISES 140**
- 3.6 EXPLORATIONS 144**
  - Advanced Pattern Matching 144

**4****OTHER LINEAR STRUCTURES 146**

- 4.1 ADT: *STACK* 146**
- 4.2 IMPLEMENTATIONS OF *STACK* 151**
  - Efficiency Issues 151
  - Stacks as a Derived Class 152
  - Stacks from Scratch 153
- 4.3 APPLICATION: POSTFIX ARITHMETIC 154**
- 4.4 ADT: *QUEUE* 157**
- 4.5 IMPLEMENTATIONS OF *QUEUE* 158**
  - Queues as Linked Lists 159
  - Circular Arrays and Queues 160
- 4.6 APPLICATION (CONTINUED): INFIX TO POSTFIX  
CONVERSION 163**
  - Verification 166
- 4.7 SUMMARY 166**
- 4.8 EXERCISES 167**

- 4.9 EXPLORATIONS 173**  
The Electronic Labyrinth 173  
Operating System Simulation 178

**PART  
THREE**

**NONLINEAR STRUCTURES**

**5**

**RECURSION 183**

- 5.1 RECURSIVE ALGORITHMS 183**  
Induction and Recursion 190
- 5.2 TIMING RECURSIVE ALGORITHMS 191**
- 5.3 COMPUTER SCIENCE INTERLUDE: DESIGN OF ALGORITHMS 196**
- 5.4 RECURSIVE DATA STRUCTURES 202**  
General Lists and LISP 204
- 5.5 SUMMARY 211**
- 5.6 EXERCISES 212**
- 5.7 EXPLORATIONS 219**  
Quicksort 219

**6**

**TREES 223**

- 6.1 THE STRUCTURE OF TREES 224**
- 6.2 ADT: BINARYTREE 228**
- 6.3 BINARY TREE TRAVERSALS 231**
- 6.4 IMPLEMENTATION OF BINARYTREE 236**
- 6.5 COMPUTER SCIENCE INTERLUDE: PARSE TREES 240**

**6.6 DATA-ORDERED BINARY TREES 242**

Binary Search Trees 244

Application: *Treesort* 251**6.7 SUMMARY 252****6.8 EXERCISES 253****6.9 EXPLORATIONS 257**

Threaded Trees 257

Preamble: Tree Applications 259

Huffman Codes 261

Tries 265

**7****SPECIALIZED TREES 268****7.1 BALANCED TREES 269**

AVL Trees 270

Efficiency and Verification 277

**7.2 B-TREES 277**

k-ary Trees, Again 278

B-Trees Explained 279

Application: External Storage 289

**7.3 SUMMARY 293****7.4 EXERCISES 294****8****GRAPHS AND DIGRAPHS 297****8.1 ADT: GRAPH 298****8.2 IMPLEMENTATIONS OF GRAPH 302**

Adjacency Matrices 302

Adjacency Lists and Edge Lists 305

**8.3 GRAPH TRAVERSALS 311**

Depth-First Traversals 311

Breadth-First Traversals 312

Spanning Trees 314

- 8.4 APPLICATION: MINIMUM SPANNING TREES 317**
- 8.5 DIRECTED GRAPHS 319**
  - Application: Cheapest Paths 320
- 8.6 COMPUTER SCIENCE INTERLUDE: COMPUTATIONAL COMPLEXITY 326**
- 8.7 SUMMARY 330**
- 8.8 EXERCISES 331**
- 8.9 EXPLORATIONS 336**
  - Topological Sorting 336
  - Counting Paths 338



## **UNORDERED COLLECTIONS 342**

- 9.1 ADT: SET 342**
- 9.2 IMPLEMENTATIONS OF SET 345**
  - Bit Vectors 345
  - Sets Represented by Lists 348
- 9.3 ADT: DICTIONARY 352**
  - Associations 352
- 9.4 HASHING 356**
  - Open Hashing 362
  - Time and Space Estimates 363
- 9.5 APPLICATION: A PROBABILISTIC SPELLING CHECKER 366**
- 9.6 ADT: PRIORITYQUEUE 369**
  - Application: Heapsort 375
- 9.7 SUMMARY 376**
- 9.8 EXERCISES 377**
- 9.9 EXPLORATIONS 380**
  - Hashing, Continued 380
  - The *DisjointSet* ADT 383

Tree Representations of *DisjointSet* 384  
Application: Minimum Spanning Trees, Revisited 389

10**TRAVESTY: PUTTING IT ALL TOGETHER 391**

- 10.1 THE PROBLEM 391**
- 10.2 THE SOLUTIONS 396**
  - Arrays 397
  - Hashing 401
  - Tries 408
  - A Guest Author 416
- 10.3 APPLICATIONS 416**
  - Reactive Keyboards 417
  - Coding, Once Again 418
- 10.4 SUMMARY 419**
- 10.5 EXERCISES 420**
- 10.6 EXPLORATIONS 422**
  - Long Strings 422

**APPENDIX A: A Pascal-C++ Dictionary 425**

- A.1 Information 426**
- A.2 Program Structure 428**
- A.3 Statements 430**
- A.4 Compound Data Types 434**
- A.5 Pointers and References 435**
- A.6 Two Sample Programs 437**

**APPENDIX B: Topics in Mathematics 444**

- B.1 Exponential and Logarithmic Functions 444**
- B.2 Induction 449**
- B.3 Counting Techniques 451**
- B.4 Exercises 455**



**APPENDIX C: Random Numbers and Simulation 459**

- C.1 Random Numbers 460
- C.2 Probability Distributions 462
- C.3 Selection Algorithms 469
- C.4 Exercises 473

**APPENDIX D: Specifications of the ADTs Used  
in the Text 475****Index 487**