

PASCAL VERSION

PROGRAMMING WITH DATA STRUCTURES

ROBERT L. KRUSE



PROGRAMMING WITH DATA STRUCTURES

Pascal Version

Robert L. Kruse

*Saint Mary's University
Halifax, Nova Scotia*



Prentice Hall

Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

KRUSE, ROBERT LEROY

Programming with data structures / Pascal version

Bibliography: p. 614

Includes index.

ISBN 0-13-729238-4

1. Electronic digital computers—Programming. 2. Data structures (Computer science) 3. Pascal (Computer program language) I. Title.

QA76.6.K774 1989

005.13'3—dc19

88-26015

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the research, development, and testing of the theory and programs in the book 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 in connection with, or arising out of, the furnishing, performance, or use of these programs.

Editorial production/supervision: *Debbie Young*

Interior design: *Dawn Stanley (Aurora Graphics)*

Cover design: *Christine Gehring-Wolf*

Page layout: *Kenny Beck*

Manufacturing buyer: *Mary Noonan*

Cover art: *Accent in Rose* (1926) by *Wassily Kandinsky* (1866-1944)

Paris, Musée National d'Art Moderne

RIRAUDON / Art Resource CRL 28807

The typesetting and color separation were done by the author using PreTeX, a preprocessor and macro package for the TeX typesetting system and POSTSCRIPT page-description language. PreTeX is a trademark of Robert L. Kruse; TeX is a trademark of the American Mathematical Society; POSTSCRIPT is a registered trademark of Adobe Systems, Inc.



© 1989 by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-729238-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

PREFACE

An apprentice carpenter may want only a hammer and a saw, but a master craftsman employs many precision tools. Computer programming likewise requires reliable tools to cope with the complexity of real applications. This book treats structured problem solving, data abstraction, and the comparative study of algorithms as fundamental tools of program design. These tools are applied to develop both data structures and software engineering principles.

The goal of programming is the construction of programs that are clear, complete, and functional. Many students, however, find difficulty in translating abstract ideas into practice. This book, therefore, takes special care in the formulation of ideas into algorithms and in the refinement of algorithms into concrete programs that can be applied to practical problems. The process of data specification and abstraction, similarly, comes before the selection of data structures and their implementations.

I believe in progressing from the concrete to the abstract, in the careful development of motivating examples, followed by the presentation of ideas in a more general form. At an early stage of their careers most students need reinforcement from seeing the immediate application of the ideas that they study, and they require the practice of writing and running programs to illustrate each important concept that they learn. This book therefore contains many samples, both short procedures and complete programs of substantial length. The exercises and programming projects, moreover, constitute an indispensable part of this book. Many of these are immediate applications of the topic under study, often requesting that programs be written and run, so that algorithms may be tested and compared. Some are larger projects, and a few are suitable for use by a group of several students working together.

Synopsis

Part I: Programming Principles

The reader of this book should have some experience in elementary Pascal programming, experience typical of a one-term introductory programming course. Part I summarizes many of the important principles of writing good programs and reviews some features of Pascal from an advanced point of view. Chapters 1 and 2 take, as an example, the problem of calculating and printing the calendar of any given year. In the context of this example, Chapter 1 reviews methods for problem solving and algorithm development, emphasizes the importance of exact specifications for subprograms, and illustrates the use of preconditions and postconditions to help ensure algorithm correctness. Chapter 2 continues the study of program development with questions of style, coding, debugging, and testing. Chapter 3 takes a fresh look at four structures provided by Pascal for data encapsulation: arrays, records, sets, and files. Its aim is not only to summarize and illustrate the syntax of these structures, but to exhibit their logical connections and plant the seeds of data abstraction.

Part II: Linear Data Structures

Part II develops the concepts of information hiding, data abstraction, and modular design. Chapter 4 studies stacks, Chapter 5 queues, and Chapter 6 lists and strings. Each of these data types is studied first as a simple concept, then in the precise specification of structure and operations as an abstract data type, then in its implementation in Pascal declarations and procedures, and finally as it is applied in complete programs.

Linked stacks, queues, and lists appear in Chapters 7 and 8. Many students will have had little or no previous experience with Pascal pointer types, so Chapter 7 carefully develops the ideas of dynamic memory allocation and linked structures, as it fully presents the necessary Pascal syntax. Linked stacks, queues, and lists are then developed as alternative implementations for abstract data types that are already familiar. The text emphasizes the importance of conforming with the specifications previously introduced for each abstract data type and maintaining the modularity of structure that allows the easy replacement of one implementation by another.

Part II develops several application programs that illustrate the methods of data abstraction and modular design. Chapter 4 presents a reverse Polish calculator that uses a stack; Chapter 5 applies queues to a program that simulates traffic patterns at a small airport; Chapter 6 develops a miniature text editor that does extensive list and string processing; and Chapter 8 outlines a group project for a program that performs calculations on polynomials represented as linked lists.

Part III: Algorithms and their Analysis

Part III broaches the comparative study and analysis of algorithms in the context of searching and sorting. With binary search as developed in Chapter 9, the student learns that vast improvements can be made over the naive methods of the introductory course. The translation of the idea of binary search into a precise algorithm, however, is fraught with danger, from which a simple algorithm verification, based on a loop invariant, provides release.

From the simple sorting methods developed in Chapter 10, the student learns that any one of several different methods can prove best in different applications.

Analysis of algorithms is therefore a worthy goal. This book, however, assumes very little mathematical preparation, and therefore takes a simple and intuitive approach to algorithm analysis. The principal tool is to draw comparison trees. The general shape and size of the tree demonstrate the differences between linear and logarithmic behavior. The big Oh notation is introduced in this part to express these differences. The study of comparison trees also leads naturally, in Chapter 11, to the introduction of binary trees as a new abstract data type.

*Part IV:
Recursion*

Recursion is a powerful tool, but one that is often misunderstood and sometimes used improperly. Some textbooks treat it as an afterthought, applying it only to trivial examples and apologizing for its alleged expense. Others give little regard to its pitfalls. I have therefore essayed to provide as balanced a treatment as possible. Whenever recursion is the natural approach it is used without hesitation, but it is neither introduced early and then ignored, nor is it applied first to problems (like linear lists or binary search) for which iterative methods are equally easy. Instead, in this text, its first use is in Chapter 11, where binary trees are developed as data structures based on ideas from linear lists, binary search, and comparison trees. Chapter 12 then provides a short, but extremely important discourse on the principles of recursion and its implementation. Chapter 13 further illustrates the importance of recursion by developing mergesort and quicksort and showing the great gains in efficiency that they provide. Chapter 14, finally, presents the examples of backtracking, lookahead in games, and compilation by recursive descent to illustrate some of the broad range of applications for recursion.

*Part V:
Further
Structures and
Algorithms*

The remaining chapters of the book collect further important topics from data structures and algorithms. Chapter 15 studies tables as structures accessed by key rather than by position. Chapter 16 further develops this idea by studying hash tables. Chapter 17 introduces graphs as mathematical models useful for problem solving and studies the ways in which they can be represented by the use of lists and of tables.

Just as Chapters 15–17 point to further study in data structures, Chapter 18 points to further study in software engineering, by introducing some of its major concerns, including problem specification and analysis, prototyping, algorithm design, refinement, verification, and analysis. These concerns are illustrated by working through an example project (CONWAY's game of Life). A simple prototype program is first developed and analyzed. This analysis leads to the development of a second program for the Life game, one based on an algorithm that is sufficiently subtle as to show the need for precise specifications and verification, and one that shows why care must be taken in the choice of data structures.

Appendices

The book concludes with three appendices. The first reviews important properties of logarithms and factorials used in algorithm analysis. The second summarizes the syntax of Pascal. The third is an annotated bibliography describing references appropriate to the topics studied in the book. Notes throughout the text urge the student to consult this bibliography for further information.

Course Structure

ACM course CS2

This book is intended primarily for a second course in computer science with one term of Pascal programming as prerequisite. The book is based on the ACM course CS2 and contains all the topics specified for this course. With the rapid development of computer science, however, this course is in continual change. This course usually includes some aspects of software engineering, some treatment of data structures and abstraction, and some survey of topics that will be studied further in more advanced courses. The degree of emphasis of each, however, differs from institution to institution. This book, therefore, contains significantly more material than can be reasonably studied in most one-term courses, so that an instructor can choose topics appropriate for any of the preceding emphases.

The core topics specified for ACM Course CS2 all appear in Chapters 1–12 of this book. A one-term course based closely on CS2 will normally include most of the content of these chapters, except for some of the algorithm analyses, verifications, and some of the example programs. The later chapters present advanced optional topics suggested for possible inclusion in CS2.

layered approach

In most of its chapters this book takes a layered approach that allows the instructor to decide easily the depth to which each topic will be studied. The fundamental topics of each chapter are developed early in the chapter. The later sections contain applications and more theoretical treatments that can be omitted with no loss of continuity. These more theoretical topics, included for the interested reader, are either not referred to again or are used only in the theoretical sections appearing at the end of later chapters.

Even if it is not covered in its entirety, this book will provide enough depth to enable interested students to continue using it as a reference in later work. It is important in any case to assign major programming projects and to allow adequate time for their completion.

Further Features

- **Chapter Previews.** Each chapter begins with an outline and a brief statement of goals and content to help the reader establish perspective.
- **Application Programs.** The text includes several large, complete programs that illustrate principles of good software design and application of the methods developed in the text. Code reading is an important skill for a programmer, but one that is often neglected in textbooks.
- **Software Diskette.** With each copy of the book is included a software diskette containing all the programs and program extracts appearing in the text. By starting from this software the student can learn many benefits of reusable programs while implementing new programming projects.
- **Programming Precepts.** Many principles of good programming are summarized with short, pithy statements that are well worth remembering.
- **Marginal Notes.** Keywords and other important concepts are highlighted in the left margin, thereby allowing the reader to locate the principal ideas of a section without delay.

- **Pointers and Pitfalls.** Each chapter of the book contains a section giving helpful hints concerning problems of program design.
- **Exercises.** Exercises appear not only at the ends of chapters but with almost every major section. These exercises help with the immediate reinforcement of the ideas of the section and develop further related ideas.
- **Projects.** Programming projects also appear in most major sections. These include simple variations of programs appearing in the text, completion of projects begun in the text, and major new projects investigating questions posed in the text.
- **Review Questions.** Each chapter concludes with simple review questions to help the student collect and summarize the principal concepts of the chapter.
- **Instructor's Supplements.** Instructors teaching from this book may obtain copies of all the following materials:
 - The *Instructor's Resource Manual* contains teaching notes on each chapter, together with complete, detailed solutions to every exercise and programming project in the text.
 - The *Transparency Masters* (several hundred in total) contain enlarged copies of almost all diagrams, specifications, program segments, and other important extracts from the text.
 - The package of *Software Diskettes* contains complete, running programs for every programming project in the text. These programs are supplied in two forms: one in standard Pascal, and one that employs Turbo Pascal units to accomplish data abstraction and information hiding as appropriate.

Acknowledgments

It is a pleasure to recognize the help of the people who have contributed in many ways to the writing and production of this book.

This book, first of all, derives part of its content from *Data Structures and Program Design*, published by Prentice Hall in 1984 and 1987. (The current book omits most of the advanced topics from that book, replacing them with additional examples and expanded explanation of more elementary topics.) My thanks, therefore, are due to the many people who have contributed to the continuing success that *Data Structures and Program Design* has enjoyed. These people—named in the preface to that book—include colleagues, students, and the editorial, marketing, and sales staff of Prentice Hall.

The writing, revision, and production of this book have been long and difficult but helped by the frequent encouragement I have received. My mother, first of all, gave me the patient understanding and love without which the work could not have been completed. Family and friends have cheered me on; colleagues have given valuable suggestions and advice; and students have shown the enthusiasm and joy of discovery that make the effort worthwhile.

ANDREW L. MEADE and J. DAVID BROWN have worked diligently and faithfully with me in producing solutions to all the exercises and programming projects, in preparing the solutions manual and the software diskettes, in testing all the programs from the book, in bringing the text files up to date, and in improving the consistency and clarity of exposition. STEVEN A. MATHESON has helped greatly with the design and programming of PreTeX, the preprocessor and macro package used to typeset this book in conjunction with DONALD KNUTH's typesetting system TeX and the page-description language POSTSCRIPT.

A good many reviewers have suggested ways to improve the organization and exposition of the book. Among these are ANDREW BERNAT (University of Texas), JOHN CUPAK (Pennsylvania State University), EILEEN ENTIN (Wentworth Institute of Technology), DAVID R. FALCONER (California State University at Fullerton), FRANK GERGELYI (New Jersey Institute of Technology), DAVID KROGER (Miami University), LAWRENCE M. LEVINE (Baruch College, City University of New York), IVAN LISS (Radford University), and EMIL C. NEU (Stevens Institute of Technology). Thanks are also due to other reviewers who did not wish to be named and to those whose contributions to *Data Structures and Program Design* have been carried into the present book.

The production of this book has given me the opportunity to meet and work with many members of the Prentice Hall staff, people who have been consistently pleasant and helpful in their dealings with me. ROB DEWEY, Marketing Manager for Computer Science and Engineering, has taken a keen interest in promoting this book. ALICE DWORKIN, Supplements Editor, has been unfailingly patient, understanding, and helpful in working to produce a comprehensive instructional package. The production editor, DEBBIE YOUNG, and the staff whose names appear on the copyright page have worked hard to expedite the publication of this book while maintaining standards of the highest quality.

But it is my editors who merit the greatest thanks: JAMES F. FEGEN, JR., who first encouraged me to undertake this project, and MARCIA J. HORTON, Editor-in-Chief for Computer Science and Engineering, who worked with me with cheerful patience and keen insight to bring it to fruition. I am proud to count Jim and Marcia not only as my respected colleagues in publishing, but even more as my friends.

ROBERT L. KRUSE

PROGRAMMING WITH DATA STRUCTURES

CONTENTS

Preface xi

Synopsis	xii
Course Structure	xiv
Further Features	xiv
Acknowledgments	xv

Prologue 1

PART I

Programming Principles 5

Chapter 1

Problem Solving 7

1.1 The Calendar Problem: A First Example	8
1.2 First Step: Specifying the Problem	8
1.3 The Calendar Problem: Specifications	12
1.4 Outlining the Method: The Overall Structure	13
1.5 The Calendar Program: Overall Structure	15
1.6 Refinement: Specifying More Detail	16

1.7 Refining the Calendar Algorithm	18
1.8 Data Abstraction	25
1.9 Names	27
1.10 Documentation and Format	29
1.11 Coding	31
Pointers and Pitfalls	37
For Further Study	38
Review Questions	38

Chapter 2

Program Development 40

2.1 Declarations	41
2.1.1 Constants	41
2.1.2 Data Types	43
2.1.3 Variables	46
2.2 Coding	49
2.2.1 The Main Program	49
2.2.2 Stubs	51
2.2.3 Input and Output	52
2.2.4 Remaining Procedures	56
2.2.5 Drivers	58
2.3 Debugging	60

2.4 Program Testing	64
Pointers and Pitfalls	68
Review Questions	69

Chapter 3

Data Packaging 70

3.1 Arrays	71
3.1.1 One-Dimensional Arrays	72
3.1.2 Multidimensional Arrays and Arrays of Arrays	74
3.2 Records	78
3.2.1 Definition and Examples	78
3.2.2 Hierarchical Records: Data Abstraction	79
3.2.3 The with Statement	80
3.2.4 Variant Records	82
3.3 Sets	85
3.4 Files	89
3.4.1 Files and Arrays	89
3.4.2 Pascal Processing with read and write	90
3.4.3 File Windows	92
3.4.4 Text Files	95
3.5 Choosing Data Structures	101
Pointers and Pitfalls	102
Review Questions	103

PART II

Linear Data Structures 105

Chapter 4

Stacks 107

4.1 Definition and Operations	108
4.1.1 Examples	110
4.1.2 Information Hiding	111
4.1.3 Specifications for a Stack	112
4.2 Array Implementation of Stacks	116
4.3 Application: Reverse Polish Calculator	121
4.4 Translation into Postfix Form	129
Pointers and Pitfalls	137
Review Questions	138

Chapter 5

Queues 139

5.1 Definitions	140
5.2 Implementations of Queues	143
5.3 Circular Queues in Pascal	149
5.3.1 Implementation with a Counter	149
5.3.2 Implementation with Special Index Values	151
5.4 Application of Queues: Simulation	154
5.4.1 Introduction	154
5.4.2 Simulation of an Airport	154
5.4.3 The Main Program	156
5.4.4 Steps of the Simulation	158
5.4.5 Random Numbers	162
5.4.6 Sample Results	164
Pointers and Pitfalls	169
Review Questions	169

Chapter 6

Lists and Strings 171

6.1 Operations on Lists	172
6.1.1 Simple Operations	172
6.1.2 Windows	173
6.1.3 List Changes	176
6.1.4 Other Operations	177
6.2 Implementation of Lists	180
6.2.1 Type Declarations	180
6.2.2 Simple Operations	180
6.2.3 Window Operations	182
6.2.4 List Changes	183
6.3 Strings	186
6.3.1 String Operations	187
6.3.2 Implementation of Strings	189
6.4 Application: A Text Editor	196
6.4.1 Specifications	196
6.4.2 Implementation	197
Pointers and Pitfalls	206
Review Questions	207

Chapter 7

Linked Stacks and Queues 208

- 7.1 Dynamic Memory Allocation and Pointers 209**
 - 7.1.1 The Problem of Overflow 209
 - 7.1.2 Pointers 209
 - 7.1.3 Further Remarks 211
- 7.2 Pointers and Dynamic Memory in Pascal 212**
- 7.3 The Basics of Linked Lists 217**
- 7.4 Linked Stacks 222**
 - 7.4.1 Declarations 222
 - 7.4.2 Pushing and Popping Entries 222
 - 7.4.3 Processing Nodes 223
 - 7.4.4 Other Operations 227
- 7.5 Linked Queues 229**
- 7.6 Abstract Data Types and Their Implementations 233**
 - 7.6.1 Introduction 233
 - 7.6.2 General Definitions 235
 - 7.6.3 Refinement of Data Specification 238
- Pointers and Pitfalls 240**
- Review Questions 241**

Chapter 8

Linked Lists 242

- 8.1 Simply Linked Implementation of Lists 243**
- 8.2 Doubly Linked Lists and Other Variations 253**
 - 8.2.1 Declarations for a Doubly Linked List 254
 - 8.2.2 Operations on Doubly Linked Lists 254
 - 8.2.3 Other Variations 256
 - 8.2.4 Comparison of Implementations 258
 - 8.2.5 Programming Hints 259
- 8.3 Application: Polynomial Arithmetic 262**
 - 8.3.1 Purpose of the Project 262
 - 8.3.2 Data Structures and Their Implementation 263
 - 8.3.3 Reading and Writing Polynomials 265
 - 8.3.4 Addition of Polynomials 267
 - 8.3.5 Completing the Project 269
- 8.4 Linked Lists in Arrays 272**
 - Pointers and Pitfalls 281**
 - Review Questions 282**

PART III**Algorithms and their Analysis 285**

Chapter 9

Searching 287

- 9.1 Searching: Introduction and Notation 288**
- 9.2 Sequential Search 290**
- 9.3 Binary Search 297**
 - 9.3.1 Algorithm Development 297
 - 9.3.2 Analysis of Binary Search 299
 - 9.3.3 Comparison with Sequential Search 303
- 9.4 Asymptotics 307**
 - 9.4.1 Introduction 307
 - 9.4.2 The Big Oh Notation 308
 - 9.4.3 Imprecision of the Big Oh Notation 311
 - 9.4.4 Ordering of Common Functions 312
- 9.5 A Lower Bound for Searching 313**
 - Pointers and Pitfalls 316**
 - Review Questions 316**

Chapter 10

Sorting 318

- 10.1 Introduction: Ordered Lists 319**
- 10.2 Insertion Sort 322**
 - 10.2.1 Ordered Insertion 322
 - 10.2.2 Sorting by Insertion 324
 - 10.2.3 Contiguous Insertion Sort 326
 - 10.2.4 Analysis 327
- 10.3 Selection Sort 331**
 - 10.3.1 The Algorithm 331
 - 10.3.2 Contiguous Implementation 332
 - 10.3.3 Analysis 334
 - 10.3.4 Comparisons 334
- 10.4 Shell Sort 336**
- 10.5 Lower Bounds 339**
 - Pointers and Pitfalls 343**
 - Review Questions 343**

PART IV

Recursion 345**Chapter 11****Divide and Conquer:
Binary Trees 347**

11.1 Examples of Recursion	348
11.1.1 Factorials: A Recursive Definition	348
11.1.2 The Towers of Hanoi	350
11.2 Binary Trees	358
11.2.1 Definitions	358
11.2.2 Traversal of Binary Trees	360
11.2.3 Linked Implementation of Binary Trees	365
11.3 Binary Search Trees	370
11.3.1 Ordered Lists and Implementations	371
11.3.2 Treesearch	372
11.3.3 Insertion into a Binary Search Tree	376
11.3.4 Treesort	381
11.3.5 Deletion from a Binary Search Tree	383
11.3.6 Implementation of Ordered Lists	385
Pointers and Pitfalls	391
Review Questions	392

Chapter 12**Principles of Recursion 393**

12.1 Designing Recursive Algorithms	394
12.2 How Recursion Works	394
12.2.1 Multiple Processors: Concurrency	395
12.2.2 Single-Processor Implementation: Storage Areas	395
12.2.3 Re-Entrant Programs	397
12.2.4 Data Structures: Stacks and Trees	397
12.2.5 Conclusion	399
12.3 Tail Recursion	399
12.4 When Not to Use Recursion	401
12.4.1 Factorials	401
12.4.2 Fibonacci Numbers	402
12.4.3 Comparisons between Recursion and Iteration	405

12.5 Guidelines and Conclusions	406
Pointers and Pitfalls	408
Review Questions	409

Chapter 13**Further Sorting Methods 410**

13.1 Divide-and-Conquer Sorting	411
13.1.1 The Main Ideas	411
13.1.2 An Example	412
13.2 Mergesort for Linked Lists	417
13.2.1 The Procedures	417
13.2.2 Analysis of Mergesort	420
13.3 Quicksort for Contiguous Lists	424
13.3.1 The Main Procedure	424
13.3.2 Partitioning the List	425
13.3.3 Analysis of Quicksort	427
13.3.4 Average-Case Analysis of Quicksort	428
13.4 Heaps and Heapsort: Contiguous Implementation of Binary Trees	436
13.4.1 Binary Trees in Contiguous Storage	437
13.4.2 Heaps and Heapsort	438
13.4.3 Analysis of Heapsort	443
13.4.4 Priority Queues	444
13.5 Review: Comparison of Methods	446
Pointers and Pitfalls	450
Review Questions	451

Chapter 14**Further Applications
of Recursion 452**

14.1 Backtracking: Postponing the Work	453
14.1.1 Solving the Eight-Queens Puzzle	453
14.1.2 Example: Four Queens	454
14.1.3 Backtracking	455
14.1.4 Refinement: Choosing the Data Structures	456
14.1.5 Analysis of Backtracking	459

14.2 Tree-Structured Programs: Look-Ahead in Games	461
14.2.1 Game Trees	462
14.2.2 The Minimax Method	463
14.2.3 Algorithm Development	465
14.2.4 Refinement	466
14.3 Compilation by Recursive Descent	469
14.3.1 The Main Program	470
14.3.2 Type Declarations	471
14.3.3 Parsing Statements	472
Pointers and Pitfalls	475
Review Questions	476

PART V

Further Structures and Algorithms 477

Chapter 15

Tables 479

15.1 Introduction: Breaking the $\lg n$ Barrier	480
15.2 Rectangular Arrays	480
15.3 Tables of Various Shapes	483
15.3.1 Triangular Tables	483
15.3.2 Jagged Tables	485
15.3.3 Inverted Tables	486
15.4 Tables: A New Abstract Data Type	489
15.5 Application: Radix Sort	492
15.5.1 The Idea	492
15.5.2 Implementation	493
15.5.3 Analysis	497
Pointers and Pitfalls	497
Review Questions	498

Chapter 16

Hash Tables 499

16.1 Sparse Tables	500
16.2 Choosing a Hash Function	502
16.3 Collision Resolution with Open Addressing	504

16.4 Collision Resolution by Chaining	511
16.5 Analysis of Hashing	515
16.6 Conclusions: Comparison of Methods	521
Pointers and Pitfalls	522
Review Questions	522

Chapter 17

Graphs 524

17.1 Mathematical Background	525
17.1.1 Definitions and Examples	525
17.1.2 Undirected Graphs	526
17.1.3 Directed Graphs	526
17.2 Computer Representation	527
17.3 Graph Traversal	532
17.3.1 Methods	532
17.3.2 Depth-First Algorithm	533
17.3.3 Breadth-First Algorithm	534
17.4 Topological Sorting	535
17.4.1 The Problem	535
17.4.2 Depth-First Algorithm	537
17.4.3 Breadth-First Algorithm	538
17.5 A Greedy Algorithm: Shortest Paths	540
17.6 Graphs as Data Structures	545
Pointers and Pitfalls	547
Review Questions	547

Chapter 18

Introduction to Software Engineering 548

18.1 The Software Life Cycle	549
18.2 The Game of Life	552
18.2.1 Rules for the Game of Life	553
18.2.2 Examples	553
18.2.3 The First Solution	555
18.2.4 Life: The First Main Program	556
18.2.5 Subprograms for Life	557
18.3 Program Review and Problem Solving	560
18.4 Algorithm Development: A Second Version of Life	565

18.5 Verification of Algorithms	570
18.5.1 Proving the Program	570
18.5.2 Invariants and Assertions	572
18.5.3 Initialization	573
18.6 Implementation and Coding	575
18.6.1 Implementation of Data Types	575
18.6.2 High-Level Procedures	577
18.6.3 Processing the Hash Table	581
18.6.4 Other Subprograms	582
18.7 Program Analysis and Comparison	583
Pointers and Pitfalls	586
Review Questions	588

Appendix A

Logarithms and Factorials 589

A.1 Applications	589
A.2 Definition and Properties	590
A.3 Choice of Base	592
A.4 Stirling's Approximation	594

Appendix B

Pascal Notes 596

B.1 Syntax Diagrams	596
B.2 General Rules	606
B.2.1 Identifiers	606
B.2.2 Rules for Spaces	607
B.2.3 Guidelines Used for Program Format	608
B.2.4 Punctuation	608
B.2.5 Alternative Symbols	609

B.3 Standard Declarations	609
B.3.1 Constants	609
B.3.2 Types	611
B.3.3 Variables	611
B.3.4 Procedures	612
B.3.5 Functions	612

B.4 Operators 613

Appendix C

References for Further Study 614

Part I. Programming Principles	614
Problem Solving	614
Program Design and Style	614
Part II. Linear Data Structures	616
Part III. Algorithms and their Analysis	617
Part IV. Recursion	618
General Expositions	618
Towers of Hanoi	618
Binary Trees	619
Sorting	619
Backtracking	620
Look-Ahead in Games	620
Compilation by Recursive Descent	620
Part V. Further Structures and Algorithms	620
Tables	620
Graphs	621
Software Engineering	621
Logarithms and Other Mathematical Methods	622

Index 623

PROLOGUE

The greatest difficulties of writing large computer programs are not in deciding what the goals of the program should be, nor even in finding methods that can be used to reach these goals. The president of a business might say, "Let's get a computer to keep track of all our inventory information, accounting records, and personnel files, and let it tell us when inventories need to be reordered and budget lines are overspent, and let it handle the payroll." With enough time and effort, a staff of systems analysts and programmers might be able to determine how various staff members are now doing these tasks and write programs to do the work in the same way.

*problems of large
programs*

This approach, however, is almost certain to be a disastrous failure. While interviewing employees, the systems analysts will find some tasks that can be put on the computer easily and will proceed to do so. Then, as they move other work to the computer, they will find that it depends on the first tasks. The output from these, unfortunately, will not be quite in the proper form. Hence they need more programming to convert the data from the form given for one task to the form needed for another. The software project begins to resemble a patchwork quilt. Some of the pieces are stronger, some weaker. Some of the pieces are carefully sewn onto the adjacent ones, some are barely tacked together. If the programmers are lucky, their creation may hold together well enough to do most of the routine work most of the time. But if any change must be made, it will have unpredictable consequences throughout the system. Later, a new request will come along, or an unexpected problem, perhaps even an emergency, and the programmers' efforts will prove as effective as using a patchwork quilt as a safety net to catch people jumping from a tall building.