

大学计算机教育丛书（影印版）

DATA STRUCTURES & PROGRAM DESIGN IN



SECOND EDITION



数据结构与程序设计

C 语言描述

第 2 版

ROBERT KRUSE / C.L. TONDO / BRUCE LEUNG



清华大学出版社 · PRENTICE HALL

<http://www.tup.tsinghua.edu.cn>

DATA STRUCTURES & PROGRAM DESIGN IN

C

Second Edition

数据结构与程序设计 (C 语言描述)

第 2 版

Robert L. Kruse

St. Mary's University Halifax, Nova Scotia Canada

Clovis L. Tondo

T&T TechWorks, Inc. Coral Springs, Florida U.S.A.

Bruce P. Leung

*Connected Components Corp. Cambridge, Massachusetts U.
S.A*

清华大学出版社

Prentice-Hall International, Inc.

(京)新登字 158 号

Data structures & program design in C 2nd ed. /Robert L. Kruse, Clovis L. Tondo, Bruce P. Leung

Copyright © 1997, 1991 by Prentice Hall, Inc.

Original English Language Edition published by Prentice Hall, Inc.

All Rights Reserved.

For sale in Mainland China only.

本书影印版由 Prentice Hall 出版公司授权清华大学出版社在中国境内(不包括中国香港、澳门特别行政区和台湾地区)独家出版发行。

本书之任何部分未经出版者书面许可,不得用任何方式复制或抄袭。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号: 01-98-1089

图书在版编目(CIP)数据

数据结构与程序设计(C语言描述)第2版:英文/克鲁泽(Kruse, R.)等著. - 影印版. - 北京:清华大学出版社,1998.7

(大学计算机教育丛书)

ISBN 7-302-02943-1

I. 数… II. 克… III. C语言-数据结构-程序设计-教材-英文
IV. TP311.12

中国版本图书馆 CIP 数据核字(98)第 09273 号

出版者: 清华大学出版社(北京清华大学学研大厦,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者: 清华大学印刷厂

发行者: 新华书店总店北京发行所

开 本: 850×1168 1/32 印张: 21.625

版 次: 1998 年 7 月第 1 版 2001 年 2 月第 3 次印刷

书 号: ISBN 7-302-02943-1/TP·1555

印 数: 10001~11500

定 价: 32.00 元

出版者的话

今天,我们的大学生、研究生和教学、科研工作者,面临的是一个国际化的信息时代。他们将需要随时查阅大量的外文资料;会有更多的机会参加国际性学术交流活动;接待外国学者;走上国际会议的讲坛。作为科技工作者,他们不仅应有与国外同行进行口头和书面交流的能力,更为重要的是,他们必须具备极强的查阅外文资料获取信息的能力。有鉴于此,在国家教委所颁布的“大学英语教学大纲”中有一条规定:专业阅读应作为必修课程开设。同时,在大纲中还规定了这门课程的学时和教学要求。有些高校除开设“专业阅读”课之外,还在某些专业课拟进行英语授课。但教、学双方都苦于没有一定数量的合适的英文原版教材作为教学参考书。为满足这方面的需要,我们陆续精选了一批国外计算机科学方面最新版本的著名教材,进行影印出版。我社获得国外著名出版公司和原著作者的授权将国际先进水平的教材引入我国高等学校,为师生们提供了教学用书,相信会对高校教材改革产生积极的影响。

我们欢迎高校师生将使用影印版教材的效果、意见反馈给我们,更欢迎国内专家、教授积极向我社推荐国外优秀计算机教育教材,以利我们将《大学计算机教育丛书(影印版)》做得更好,更适合高校师生的需要。

清华大学出版社
《大学计算机教育丛书(影印版)》项目组

1999.6

Preface

An apprentice carpenter may want only a hammer and a saw, but a master craftsman employs many precision tools. Computer programming likewise requires sophisticated tools to cope with the complexity of real applications, and only practice with these tools will build skill in their use. This book treats structured problem solving, data abstraction, software engineering principles, and the comparative analysis of algorithms as fundamental tools of program design. Several case studies of substantial size are worked out in detail, to show how all the tools are used together to build complete programs.

Many of the algorithms and data structures we study possess an intrinsic elegance, a simplicity that cloaks the range and power of their applicability. Before long the student discovers that vast improvements can be made over the naïve methods usually used in introductory courses. Yet this elegance of method is tempered with uncertainty. The student soon finds that it can be far from obvious which of several approaches will prove best in particular applications. Hence comes an early opportunity to introduce truly difficult problems of both intrinsic interest and practical importance and to exhibit the applicability of mathematical methods to algorithm verification and analysis.

Many students 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.

We 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 sample programs, both short functions and complete programs of substantial length. The exercises and programming projects, moreover, constitute an indispensable part of the 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 small group of students working together.

Synopsis

1. *Programming Principles*

By working through the first large project (CONWAY's game of Life), Chapter 1 expounds principles of top-down refinement, program design, review, and testing, principles that the student will see demonstrated and is expected to follow throughout the sequel. At the same time, this project provides an opportunity for the student to review the syntax of C, the programming language used throughout the book.

2. *Introduction to Software Engineering*

Chapter 2 introduces a few of the basic concerns of software engineering, including problem specification and analysis, prototyping, data abstraction, algorithm design, refinement, verification, and analysis. The chapter applies these principles 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.

3. *Stacks and Recursion*

Chapter 3 continues to elucidate data abstraction and algorithm design by studying stacks as an abstract data type, recursion as a problem-solving method, and the intimate connections among stacks, recursion, and certain trees.

4. *Queues and Linked Lists*

Queues and lists are the central topics of the next two chapters. The chapters expound several different implementations of each abstract data type, develop large application programs showing the relative advantages of different implementations, and introduce algorithm analysis in a very informal way. A major goal of these chapters is to bring the student to appreciate data abstraction and to apply methods of top-down design to data as well as to algorithms.

5. *General Lists*

6. *Searching*

Chapters 6, 7, and 8 present algorithms for searching, sorting, and table access (including hashing). These chapters illustrate the interplay between algorithms and the associated abstract data types, data structures, and implementations. The text introduces the "big O" notation for elementary algorithm analysis and highlights the crucial choices to be made regarding best use of space, time, and programming effort.

7. *Sorting*

8. *Tables and Information Retrieval*

These choices require that we find analytical methods to assess algorithms, and producing such analyses is a battle for which combinatorial mathematics must provide the arsenal. At an elementary level we can expect students neither to be well armed nor to possess the mathematical maturity needed to hone their skills to perfection. Our goal, therefore, is to help students recognize the importance of such skills in anticipation of later chances to study mathematics.

9. *Binary Trees*

Binary trees are surely among the most elegant and useful of data structures. Their study, which occupies Chapter 9, ties together concepts from lists, searching, and sorting. As recursively defined data structures, binary trees afford an excellent opportunity for the student to become comfortable with recursion applied both to data structures and algorithms. The chapter begins with elementary topics and progresses as far as splay trees and amortized algorithm analysis.

10. *Multitway Trees*

11. *Graphs*

Chapter 10 continues the study of more sophisticated data structures, including tries, B-trees, and red-black trees. The next chapter introduces graphs as more general structures useful for problem solving.

12. *Case Study: The Polish Notation*

The case study in Chapter 12 examines the Polish notation in considerable detail, exploring the interplay of recursion, trees, and stacks as vehicles for problem

solving and algorithm development. Some of the questions addressed can serve as an informal introduction to compiler design. As usual, the algorithms are fully developed within a functioning C program. This program accepts as input an expression in ordinary (infix) form, translates the expression into postfix form, and evaluates the expression for specified values of the variable(s).

The appendices discuss several topics that are not properly part of the book's subject but that are often missing from the student's preparation.

*A. Mathematical
Methods*

Appendix A presents several topics from discrete mathematics. Its final two sections, on Fibonacci and Catalan numbers, are more advanced and not needed for any vital purpose in the text, but are included to encourage combinatorial interest in the more mathematically inclined.

*B. Removal of
Recursion*

Removal of recursion is a topic that most programmers should no longer need to study. But at present some important work must still be done in contexts (like FORTRAN or COBOL) disallowing recursion. Methods for manual recursion removal are therefore sometimes required, and are collected for reference as Appendix B. Some instructors will wish to include the study of threaded binary trees with Chapter 9; this section is therefore written so that it can be read independently of the remainder of Appendix B.

*C. An Introduction
to C*

Appendix C, finally, is a brief introduction to the C programming language. This is not a thorough treatment of the language, but it is intended to serve as a review of C syntax and as a reference for the student.

Changes in the Second Edition

In this edition, the entire text has been carefully reviewed and revised to update its presentation and to reflect the ideas of many readers who have communicated their experiences in studying the book. The principal changes are summarized as follows.

- All the programs have been rewritten, revised, and polished to emphasize data abstraction, to develop and employ reusable code, and to strengthen uniformity and elegance of style.
- The documentation has been strengthened by including informal specifications (pre- and postconditions) with all subprograms.
- Recursion is treated much earlier in the text and then emphasized by repeated use thereafter.
- The coverage of more advanced, modern topics has been extended by the inclusion of several new sections, including splay trees, red-black trees, and amortized algorithm analysis.
- The text highlights new case studies, such as the miniature text editor in Chapter 5.
- New exercises and programming projects have been added, including continuing projects on information retrieval that request the student to compare the performance of several different data structures and algorithms.

- The material on graph theory and graph algorithms has now been collected as a separate chapter.
- The treatment of lists has been streamlined.
- The source code for all the programs and program extracts printed in the book will be available on the internet. To reach this software under ftp, log in as user anonymous on the ftp site `prehnall.com` and change to the directory
`pub/esm/computer.science.s-041/kruse/dspdc2`
- Instructors teaching from this book may obtain, at no charge, the *Instructor's Resource Manual*, which includes:
 - Brief teaching notes on each chapter;
 - Full solutions to all exercises in the textbook;
 - Transparency masters;
 - A PC disk containing both the software mentioned previously and the full source code for all programming projects from the textbook.

Course Structure

prerequisite

The prerequisite for this book is a first course in programming, with experience using the elementary features of C. Appendix C presents several advanced aspects of C programming that are often omitted from introductory courses. A good knowledge of high school mathematics will suffice for almost all the algorithm analyses, but further (perhaps concurrent) preparation in discrete mathematics will prove valuable. Appendix A reviews all required mathematics.

content

This book is intended for courses such as the ACM Course CS2 (*Program Design and Implementation*), ACM Course CS7 (*Data Structures and Algorithm Analysis*), or a course combining these. Thorough coverage is given to most of the ACM/IEEE knowledge units¹ on data structures and algorithms. These include:

- AL1 Basic data structures, such as arrays, tables, stacks, queues, trees, and graphs;
- AL2 Abstract data types;
- AL3 Recursion and recursive algorithms;
- AL4 Complexity analysis using the big O notation;
- AL6 Sorting and searching; and
- AL8 Practical problem-solving strategies, with large case studies.

The three most advanced knowledge units, AL5 (complexity classes, NP-complete problems), AL7 (computability and undecidability), and AL9 (parallel and distributed algorithms) are not treated in this book.

Most chapters of this book are structured so that the core topics are presented first, followed by examples, applications, and larger case studies. Hence, if time allows only a brief study of a topic, it is possible, with no loss of continuity, to move

¹ See *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM Press, New York, 1990.

rapidly from chapter to chapter covering only the core topics. When time permits, however, both students and instructor will enjoy the occasional excursion into the supplementary topics and worked-out projects.

two-term course

A two-term course can cover nearly the entire book, thereby attaining a satisfying integration of many topics from the areas of problem solving, data structures, program development, and algorithm analysis. Students need time and practice to understand general methods. By combining the studies of data abstraction, data structures, and algorithms with their implementations in projects of realistic size, an integrated course can build a solid foundation on which, later, more theoretical courses can be built.

Even if this book is not covered in its entirety, it 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.

Book Production

This book and its supplements were written and produced with the first author's software called `PreTeX`, a preprocessor and macro package for the `TEX` typesetting system.² `PreTeX`, by exploiting context dependency, automatically supplies much of the typesetting markup required by `TEX`. `PreTeX` also supplies several tools useful to the author, such as a powerful cross-reference system, greatly simplified typesetting of mathematics and computer-program listings, and automatic generation of the index and table of contents, while allowing the processing of the book in conveniently small files at every stage. Solutions, placed with exercises and projects, are automatically removed from the text and placed in a separate manual. In conjunction with the `PostScript` page-description language, `PreTeX` provides convenient facilities for color separation, halftone screens, and other special results.

For a book such as this, `PreTeX`'s treatment of computer programs is its most important feature. Computer programs are not included with the main body of the text; instead, they are placed in separate, secondary files, along with any desired explanatory text, and with any desired typesetting markup in place. By placing tags at appropriate places in the secondary files, `PreTeX` can extract arbitrary parts of a secondary file, in any desired order, for typesetting with the text. Another utility (called `StripTeX`) can be used on the same file to remove all the tags, text, and markup, with output that is a program ready to be compiled. The same input file thus automatically produces both typeset program listings and compiled program code. In this way, the reader gains increased confidence in the accuracy of the computer program listings appearing in the text.

For this edition, all the diagrams and artwork have been produced as `PostScript` code in Adobe Illustrator. This allows the automatic inclusion of all figures in the preliminary drafts of the manuscript and shortens the final stages of production by removing any need for manual processing of camera copy.

² `TEX` was developed by DONALD E. KNUTH, who has also made many important contributions to our knowledge of data structures and algorithms. (See the entries under his name in the index.)

Acknowledgments

Over the years, this book and its Pascal antecedents have benefitted greatly from the contributions of many people: family, friends, colleagues, and students. The first edition lists some of the people whose contributions are especially noteworthy. Since the publication of the first edition, translations into several languages have also appeared, and many more people have kindly forwarded their comments and suggestions to us. In particular, it is a pleasure to acknowledge the suggestions of the reviewers for the current edition: ALEX RYBA (Marquette University), RICHARD SAUNDERS (University of Arizona), DAVID STRAIGHT (University of Tennessee, Knoxville), CARLOS CUNHA (Boston University), and GREG CAMERON (Ricks College).

C. L. TONDO also acknowledges the help of GEORGE EDMUNDS, TOM HORTON, SAM HSU, MARIA PETRIE (all of Florida Atlantic University), ROUE GUILD (Nova Southeastern University), LOUIS VOSLOO (Y&Y, Inc.), NELSON FELIPPE DA SILVA (Polydata, Inc.), LUIZ BIAVATTI, A. CARLOS TONDO (T&T TechWorks, Inc.), ED HAUGHNEY, ANDREW NATHANSON, RED VISCUSO, and CAREN E. TONDO.

The editorial staff of Prentice Hall, especially ALAN APT, Publisher, and LAURA STEELE, Managing Editor, have displayed much patience, interest, and helpfulness in bringing this project to a successful conclusion.

JIM COOPER of PreTeX, Inc., has expedited the appearance of this book and its supplements by checking all the C programs, solving many problems of page makeup, and by completing all the solutions to exercises and reworking the programming projects.

Finally, let us note that this book is an adaptation into C, by the second and third authors, of the Pascal-based *Data Structures and Program Design*, third edition, by the first author. The first author is responsible for the language-independent discussion and the other authors for the C programs and language-specific exposition.

ROBERT L. KRUSE
CLOVIS L. TONDO
BRUCE P. LEUNG

Contents

| | |
|-------------------------------|------|
| PREFACE | xi |
| Synopsis | xii |
| Changes in the Second Edition | xiii |
| Course Structure | xiv |
| Book Production | xv |
| Acknowledgments | xvi |

CHAPTER 1

Programming Principles _____ 1

| | |
|---|----|
| 1.1 Introduction | 2 |
| 1.2 The Game of Life | 4 |
| 1.2.1 Rules for the Game of Life | 4 |
| 1.2.2 Examples | 5 |
| 1.2.3 The Solution | 6 |
| 1.2.4 Life: The Main Program | 7 |
| 1.3 Programming Style | 10 |
| 1.3.1 Names | 10 |
| 1.3.2 Documentation and Format | 12 |
| 1.3.3 Refinement and Modularity | 14 |
| 1.4 Coding, Testing, and Further Refinement | 19 |
| 1.4.1 Stubs | 19 |
| 1.4.2 Counting Neighbors | 20 |
| 1.4.3 Input and Output | 21 |
| 1.4.4 Drivers | 24 |
| 1.4.5 Program Tracing | 25 |
| 1.4.6 Principles of Program Testing | 26 |

Pointers and Pitfalls 30

Review Questions 32

| | |
|------------------------------|----|
| References for Further Study | 32 |
| C | 32 |
| Programming Principles | 33 |
| The Game of Life | 33 |

CHAPTER 2

Introduction to Software Engineering _____ 34

| | |
|---|----|
| 2.1 Program Maintenance | 35 |
| 2.1.1 Review of the Life Program | 35 |
| 2.1.2 A Fresh Start and a New Method for Life | 37 |

| | |
|---|----|
| 2.2 Algorithm Development: A Second Version of Life | 40 |
| 2.2.1 Lists: Specifications for a Data Structure | 40 |
| 2.2.2 The Main Program | 45 |
| 2.2.3 Information Hiding | 47 |
| 2.2.4 Refinement: Development of the Subprograms | 48 |
| 2.2.5 Verification of Algorithms | 50 |

| | |
|---------------------------------|----|
| 2.3 Coding | 55 |
| 2.3.1 The List Functions | 55 |
| 2.3.2 Error Processing | 56 |
| 2.3.3 Demonstration and Testing | 57 |

2.4 Coding the Life Functions 62

2.5 Program Analysis and Comparison 66

| | |
|-----------------------------|----|
| 2.6 Conclusions and Preview | 68 |
| 2.6.1 The Game of Life | 68 |
| 2.6.2 Program Design | 70 |
| 2.6.3 C | 73 |

Pointers and Pitfalls 75

Review Questions 75

References for Further Study 76

| | |
|--|------------|
| CHAPTER 3 | |
| Stacks and Recursion | 77 |
| 3.1 Stacks | 78 |
| 3.1.1 Introduction | 78 |
| 3.1.2 First Example: Reversing a Line | 79 |
| 3.1.3 Information Hiding | 80 |
| 3.1.4 Specifications for a Stack | 81 |
| 3.1.5 Implementation of Stacks | 83 |
| 3.1.6 Linked Stacks | 85 |
| 3.2 Introduction to Recursion | 91 |
| 3.2.1 Stack Frames for Subprograms | 91 |
| 3.2.2 Tree of Subprogram Calls | 91 |
| 3.2.3 Factorials: A Recursive Definition | 93 |
| 3.2.4 Divide and Conquer: The Towers of Hanoi | 95 |
| 3.3 Backtracking: Postponing the Work | 101 |
| 3.3.1 Solving the Eight-Queens Puzzle | 102 |
| 3.3.2 Example: Four Queens | 102 |
| 3.3.3 Backtracking | 103 |
| 3.3.4 Refinement: Choosing the Data Structures | 104 |
| 3.3.5 Analysis of Backtracking | 107 |
| 3.4 Principles of Recursion | 110 |
| 3.4.1 Designing Recursive Algorithms | 110 |
| 3.4.2 How Recursion Works | 111 |
| 3.4.3 Tail Recursion | 115 |
| 3.4.4 When Not to Use Recursion | 116 |
| 3.4.5 Guidelines and Conclusions | 120 |
| Pointers and Pitfalls | 122 |
| Review Questions | 124 |
| References for Further Study | 124 |
| CHAPTER 4 | |
| Queues and Linked Lists | 126 |
| 4.1 Definitions | 127 |
| 4.2 Implementations of Queues | 131 |
| 4.3 Circular Queues in C | 135 |
| 4.4 Application of Queues: Simulation | 139 |
| 4.4.1 Introduction | 139 |
| 4.4.2 Simulation of an Airport | 140 |
| 4.4.3 The Main Program | 142 |
| 4.4.4 Steps of the Simulation | 144 |
| 4.4.5 Pseudo-Random Numbers | 147 |
| 4.4.6 Sample Results | 149 |
| 4.5 Pointers and Linked Lists | 152 |
| 4.5.1 Introduction and Survey | 152 |
| 4.5.2 Pointers and Dynamic Memory in C | 155 |
| 4.5.3 The Basics of Linked Lists | 159 |
| 4.6 Linked Queues | 161 |
| 4.7 Application: | |
| Polynomial Arithmetic | 166 |
| 4.7.1 Purpose of the Project | 166 |
| 4.7.2 The Main Program | 166 |
| 4.7.3 Data Structures and Their Implementation | 171 |
| 4.7.4 Reading and Writing Polynomials | 172 |
| 4.7.5 Addition of Polynomials | 174 |
| 4.7.6 Completing the Project | 176 |
| 4.8 Abstract Data Types and Their Implementations | 179 |
| 4.8.1 Introduction | 179 |
| 4.8.2 General Definitions | 180 |
| 4.8.3 Refinement of Data Specification | 183 |
| Pointers and Pitfalls | 185 |
| Review Questions | 185 |
| References for Further Study | 186 |
| CHAPTER 5 | |
| General Lists | 187 |
| 5.1 List Specifications | 188 |
| 5.2 Implementation of Lists | 190 |
| 5.2.1 Contiguous Implementation | 190 |
| 5.2.2 Simply Linked Implementation | 191 |
| 5.2.3 Variation: Keeping the Current Position | 195 |
| 5.2.4 Doubly Linked Lists | 197 |
| 5.2.5 Comparison of Implementations | 200 |
| 5.3 Strings | 202 |
| 5.4 Application: A Text Editor | 205 |
| 5.4.1 Specifications | 205 |
| 5.4.2 Implementation | 207 |
| 5.5 Linked Lists in Arrays | 214 |
| 5.6 Generating Permutations | 223 |
| Pointers and Pitfalls | 228 |
| Review Questions | 229 |
| References for Further Study | 230 |

| | |
|---|------------|
| CHAPTER 6 | |
| Searching | 231 |
| 6.1 Searching: | |
| Introduction and Notation | 232 |
| 6.2 Sequential Search | 235 |
| 6.3 Coatrooms: A Project | 241 |
| 6.3.1 Introduction and Specification | 241 |
| 6.3.2 Demonstration and Testing Programs | 244 |
| 6.4 Binary Search | 248 |
| 6.4.1 Algorithm Development | 249 |
| 6.4.2 The Forgetful Version | 249 |
| 6.4.3 Recognizing Equality | 252 |
| 6.5 Comparison Trees | 254 |
| 6.5.1 Analysis for $n = 10$ | 255 |
| 6.5.2 Generalization | 258 |
| 6.5.3 Comparison of Methods | 261 |
| 6.5.4 A General Relationship | 263 |
| 6.6 Lower Bounds | 264 |
| 6.7 Asymptotics | 269 |
| 6.7.1 Introduction | 269 |
| 6.7.2 The Big-O Notation | 270 |
| 6.7.3 Imprecision of the Big-O Notation | 273 |
| 6.7.4 Ordering of Common Functions | 274 |
| Pointers and Pitfalls | 275 |
| Review Questions | 276 |
| References for Further Study | 276 |
| CHAPTER 7 | |
| Sorting | 278 |
| 7.1 Introduction and Notation | 279 |
| 7.2 Insertion Sort | 280 |
| 7.2.1 Ordered Lists | 280 |
| 7.2.2 Sorting by Insertion | 281 |
| 7.2.3 Linked Version | 283 |
| 7.2.4 Analysis | 285 |
| 7.3 Selection Sort | 288 |
| 7.3.1 The Algorithm | 289 |
| 7.3.2 Contiguous Implementation | 290 |
| 7.3.3 Analysis | 291 |
| 7.3.4 Comparisons | 291 |
| 7.4 Shell Sort | 293 |
| 7.5 Lower Bounds | 295 |
| 7.6 Divide-and-Conquer Sorting | 298 |
| 7.6.1 The Main Ideas | 298 |
| 7.6.2 An Example | 299 |
| 7.7 Mergesort for Linked Lists | 304 |
| 7.7.1 The Functions | 304 |
| 7.7.2 Analysis of Mergesort | 306 |
| 7.8 Quicksort for Contiguous Lists | 311 |
| 7.8.1 The Main Function | 311 |
| 7.8.2 Partitioning the List | 312 |
| 7.8.3 Analysis of Quicksort | 314 |
| 7.8.4 Average-Case Analysis of Quicksort | 316 |
| 7.8.5 Comparison with Mergesort | 318 |
| 7.9 Heaps and Heapsort | 321 |
| 7.9.1 Two-Way Trees as Lists | 322 |
| 7.9.2 Heapsort | 323 |
| 7.9.3 Analysis of Heapsort | 327 |
| 7.9.4 Priority Queues | 328 |
| 7.10 Review: Comparison of Methods | 330 |
| Pointers and Pitfalls | 333 |
| Review Questions | 334 |
| References for Further Study | 334 |
| CHAPTER 8 | |
| Tables and Information Retrieval | 336 |
| 8.1 Introduction: | |
| Breaking the $\lg n$ Barrier | 337 |
| 8.2 Rectangular Arrays | 337 |
| 8.3 Tables of Various Shapes | 340 |
| 8.3.1 Triangular Tables | 340 |
| 8.3.2 Jagged Tables | 342 |
| 8.3.3 Inverted Tables | 342 |
| 8.4 Tables: A New Abstract Data Type | 345 |
| 8.5 Application: Radix Sort | 348 |
| 8.5.1 The Idea | 348 |
| 8.5.2 Implementation | 349 |
| 8.5.3 Analysis | 352 |
| 8.6 Hashing | 353 |
| 8.6.1 Sparse Tables | 353 |
| 8.6.2 Choosing a Hash Function | 355 |
| 8.6.3 Collision Resolution with Open Addressing | 357 |
| 8.6.4 Collision Resolution by Chaining | 362 |
| 8.7 Analysis of Hashing | 367 |

| | |
|--|------------|
| 8.8 Conclusions: | |
| Comparison of Methods | 373 |
| 8.9 Application: | |
| The Life Game Revisited | 374 |
| 8.9.1 Choice of Algorithm | 374 |
| 8.9.2 Specification of Data Structures | 374 |
| 8.9.3 The Main Program | 376 |
| 8.9.4 Functions | 377 |
| Pointers and Pitfalls | 380 |
| Review Questions | 381 |
| References for Further Study | 382 |
| CHAPTER 9 | |
| Binary Trees | 383 |
| 9.1 Introduction to Binary Trees | 384 |
| 9.1.1 Definitions | 384 |
| 9.1.2 Traversal of Binary Trees | 386 |
| 9.1.3 Linked Implementation of Binary Trees | 391 |
| 9.2 Binary Search Trees | 396 |
| 9.2.1 Ordered Lists and Implementations | 397 |
| 9.2.2 Treesearch | 398 |
| 9.2.3 Insertion into a Binary Search Tree | 401 |
| 9.2.4 Treesort | 404 |
| 9.2.5 Deletion from a Binary Search Tree | 406 |
| 9.3 Building a Binary Search Tree | 414 |
| 9.3.1 Getting Started | 415 |
| 9.3.2 Declarations and the Main Function | 416 |
| 9.3.3 Inserting a Node | 417 |
| 9.3.4 Finishing the Task | 418 |
| 9.3.5 Evaluation | 419 |
| 9.3.6 Random Search Trees and Optimality | 419 |
| 9.4 Height Balance: AVL Trees | 422 |
| 9.4.1 Definition | 423 |
| 9.4.2 Insertion of a Node | 424 |
| 9.4.3 Deletion of a Node | 431 |
| 9.4.4 The Height of an AVL Tree | 433 |
| 9.5 Splay Trees: | |
| A Self-Adjusting Data Structure | 438 |
| 9.5.1 Introduction | 438 |
| 9.5.2 Splaying Steps | 439 |
| 9.5.3 Splaying Algorithm | 442 |
| 9.5.4 Amortized Algorithm Analysis: Introduction | 446 |
| 9.5.5 Amortized Analysis of Splaying | 449 |

| | |
|------------------------------|-----|
| Pointers and Pitfalls | 455 |
| Review Questions | 456 |
| References for Further Study | 458 |

| | |
|--|------------|
| CHAPTER 10 | |
| Multiway Trees | 459 |
| 10.1 Orchards, Trees, and Binary Trees | 460 |
| 10.1.1 On the Classification of Species | 460 |
| 10.1.2 Ordered Trees | 461 |
| 10.1.3 Forests and Orchards | 463 |
| 10.1.4 The Formal Correspondence | 464 |
| 10.1.5 Rotations | 465 |
| 10.1.6 Summary | 466 |
| 10.2 Lexicographic Search Trees: Tries | 468 |
| 10.2.1 Tries | 468 |
| 10.2.2 Searching for a Key | 468 |
| 10.2.3 C Algorithm | 470 |
| 10.2.4 Insertion into a Trie | 470 |
| 10.2.5 Deletion from a Trie | 472 |
| 10.2.6 Assessment of Tries | 472 |
| 10.3 External Searching: B-Trees | 473 |
| 10.3.1 Access Time | 473 |
| 10.3.2 Multiway Search Trees | 474 |
| 10.3.3 Balanced Multiway Trees | 474 |
| 10.3.4 Insertion into a B-tree | 475 |
| 10.3.5 C Algorithms: Searching and Insertion | 477 |
| 10.3.6 Deletion from a B-tree | 483 |
| 10.4 Red-Black Trees | 492 |
| 10.4.1 Introduction | 492 |
| 10.4.2 Definition and Analysis | 493 |
| 10.4.3 Insertion | 495 |
| 10.4.4 C Insertion | 496 |
| 10.5 Tree-Structured Programs: | |
| Look-Ahead in Games | 501 |
| 10.5.1 Game Trees | 501 |
| 10.5.2 The Minimax Method | 502 |
| 10.5.3 Algorithm Development | 503 |
| 10.5.4 Refinement | 504 |
| Pointers and Pitfalls | 507 |
| Review Questions | 507 |
| References for Further Study | 508 |

CHAPTER 11 Graphs _____ 510

- 11.1 Mathematical Background 511
 - 11.1.1 Definitions and Examples 511
 - 11.1.2 Undirected Graphs 512
 - 11.1.3 Directed Graphs 512

11.2 Computer Representation 513

- 11.3 Graph Traversal 517
 - 11.3.1 Methods 517
 - 11.3.2 Depth-First Algorithm 518
 - 11.3.3 Breadth-First Algorithm 519

- 11.4 Topological Sorting 520
 - 11.4.1 The Problem 520
 - 11.4.2 Depth-First Algorithm 522
 - 11.4.3 Breadth-First Algorithm 523

11.5 A Greedy Algorithm: Shortest Paths 525

11.6 Graphs as Data Structures 529

Pointers and Pitfalls 531

Review Questions 532

References for Further Study 532

CHAPTER 12 Case Study: The Polish Notation _____ 533

- 12.1 The Problem 534
 - 12.1.1 The Quadratic Formula 534

- 12.2 The Idea 536
 - 12.2.1 Expression Trees 536
 - 12.2.2 Polish Notation 538
 - 12.2.3 C Method 539

12.3 Evaluation of Polish Expressions 540

- 12.3.1 Evaluation of an Expression in Prefix Form 540
- 12.3.2 C Conventions 541
- 12.3.3 C Function for Prefix Evaluation 542
- 12.3.4 Evaluation of Postfix Expressions 542
- 12.3.5 Proof of the Program: Counting Stack Entries 544
- 12.3.6 Recursive Evaluation of Postfix Expressions 547

12.4 Translation from Infix Form to Polish Form 551

12.5 An Interactive Expression

- Evaluator 558
 - 12.5.1 Overall Structure 558
 - 12.5.2 Representation of the Data 560
 - 12.5.3 Initialization and Auxiliary Tasks 562
 - 12.5.4 Translation of the Expression 566
 - 12.5.5 Evaluating the Expression 574
 - 12.5.6 Graphing the Expression 576

References for Further Study 578

APPENDIX A Mathematical Methods _____ 579

A.1 Sums of Powers of Integers 580

- A.2 Logarithms 582
 - A.2.1 Definition of Logarithms 583
 - A.2.2 Simple Properties 583
 - A.2.3 Choice of Base 584
 - A.2.4 Natural Logarithms 584
 - A.2.5 Notation 585
 - A.2.6 Change of Base 586
 - A.2.7 Logarithmic Graphs 586
 - A.2.8 Harmonic Numbers 588

- A.3 Permutations, Combinations, Factorials 589
 - A.3.1 Permutations 589
 - A.3.2 Combinations 589
 - A.3.3 Factorials 590

A.4 Fibonacci Numbers 592

- A.5 Catalan Numbers 594
 - A.5.1 The Main Result 594
 - A.5.2 The Proof by One-to-One Correspondences 594
 - A.5.3 History 596
 - A.5.4 Numerical Results 597

References for Further Study 598

APPENDIX B Removal of Recursion _____ 600

- B.1 General Methods for Removing Recursion 601
 - B.1.1 Preliminary Assumptions 601
 - B.1.2 General Rules 602
 - B.1.3 Indirect Recursion 603
 - B.1.4 Towers of Hanoi 603
 - B.1.5 Further Simplifications 605

| | |
|---|------------|
| B.2 Recursion Removal by Folding | 606 |
| B.2.1 Program Schemata | 606 |
| B.2.2 Proof of the Transformation | 607 |
| B.2.3 Towers of Hanoi: The Final Version | 609 |
| B.3 Nonrecursive Quicksort | 611 |
| B.4 Stackless Recursion Removal: Mergesort | 613 |
| B.5 Threaded Binary Trees | 617 |
| B.5.1 Introduction | 617 |
| B.5.2 Threads | 619 |
| B.5.3 Inorder and Preorder Traversal | 620 |
| B.5.4 Insertion in a Threaded Tree | 621 |
| B.5.5 Postorder Traversal | 623 |
| References for Further Study | 627 |
| APPENDIX C | |
| An Introduction to C | 628 |
| C.1 Introduction | 629 |
| C.1.1 Overview of a C Program | 629 |
| C.2 C Elements | 629 |
| C.2.1 Reserved Words | 629 |
| C.2.2 Constants | 629 |
| C.3 Types and Declarations | 631 |
| C.3.1 Basic Types | 631 |
| C.3.2 Arrays | 631 |
| C.3.3 Enumerations | 631 |
| C.3.4 Structures | 632 |
| C.3.5 Unions | 632 |
| C.3.6 Type Definitions with <code>typedef</code> | 634 |

| | |
|--|------------|
| C.4 Operators | 635 |
| C.5 Control Flow Statements | 636 |
| C.5.1 If - Else | 636 |
| C.5.2 Switch | 636 |
| C.5.3 Loops | 637 |
| C.5.4 Break and Continue | 637 |
| C.5.5 Goto | 637 |
| C.6 Pointers | 638 |
| C.6.1 Pointer to a Simple Variable | 638 |
| C.6.2 Pointer to an Array | 639 |
| C.6.3 Array of Pointers | 640 |
| C.6.4 Pointer to Structures | 641 |
| C.7 Functions | 642 |
| C.7.1 Arguments to Functions: Call by Value | 642 |
| C.7.2 Arguments to Functions: Call by Reference | 643 |
| C.7.3 Function Prototypes and Include Files | 644 |
| C.8 Pointers and Functions | 644 |
| C.8.1 Pointer to a Function | 644 |
| C.8.2 Functions that Return a Pointer | 645 |
| C.8.3 Pointer to a Pointer as an Argument | 646 |
| References for Further Study | 647 |

| | |
|--------------|------------|
| INDEX | 649 |
|--------------|------------|

1

Programming Principles

THIS CHAPTER summarizes important principles of good programming, especially as applied to large projects, and illustrates methods for discovering effective algorithms. In the process we raise questions in program design that we shall address in later chapters, and review many of the special features of the language C by using them to write programs.

| | |
|--|-----------|
| 1.1 Introduction | 2 |
| 1.2 The Game of Life | 4 |
| 1.2.1 Rules for the Game of Life | 4 |
| 1.2.2 Examples | 5 |
| 1.2.3 The Solution | 6 |
| 1.2.4 Life: The Main Program | 7 |
| 1.3 Programming Style | 10 |
| 1.3.1 Names | 10 |
| 1.3.2 Documentation and Format | 12 |
| 1.3.3 Refinement and Modularity | 14 |
| 1.4 Coding, Testing, and Further Refinement | 19 |
| 1.4.1 Stubs | 19 |
| 1.4.2 Counting Neighbors | 20 |
| 1.4.3 Input and Output | 21 |
| 1.4.4 Drivers | 24 |
| 1.4.5 Program Tracing | 25 |
| 1.4.6 Principles of Program Testing | 26 |
| Pointers and Pitfalls | 30 |
| Review Questions | 32 |
| References for Further Study | 32 |
| C | 32 |
| Programming Principles | 33 |
| The Game of Life | 33 |