大学计算机教育丛书（影印版）

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 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

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

5. General 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.

6. Searching

7. Sorting

8. Tables and Information Retrieval

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.

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. Multiway 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 `prenhall.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 units[1] 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), ROLIE 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

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.