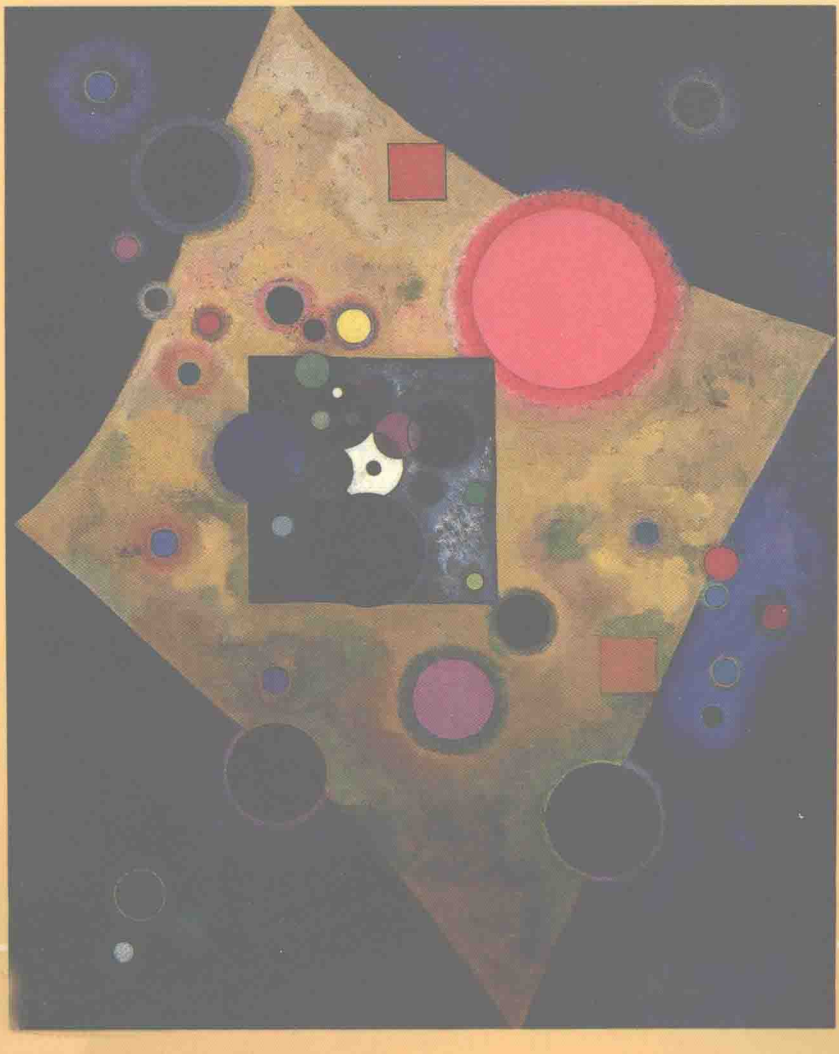
PASCAL VERSION

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

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.

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*
Pentice-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

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.