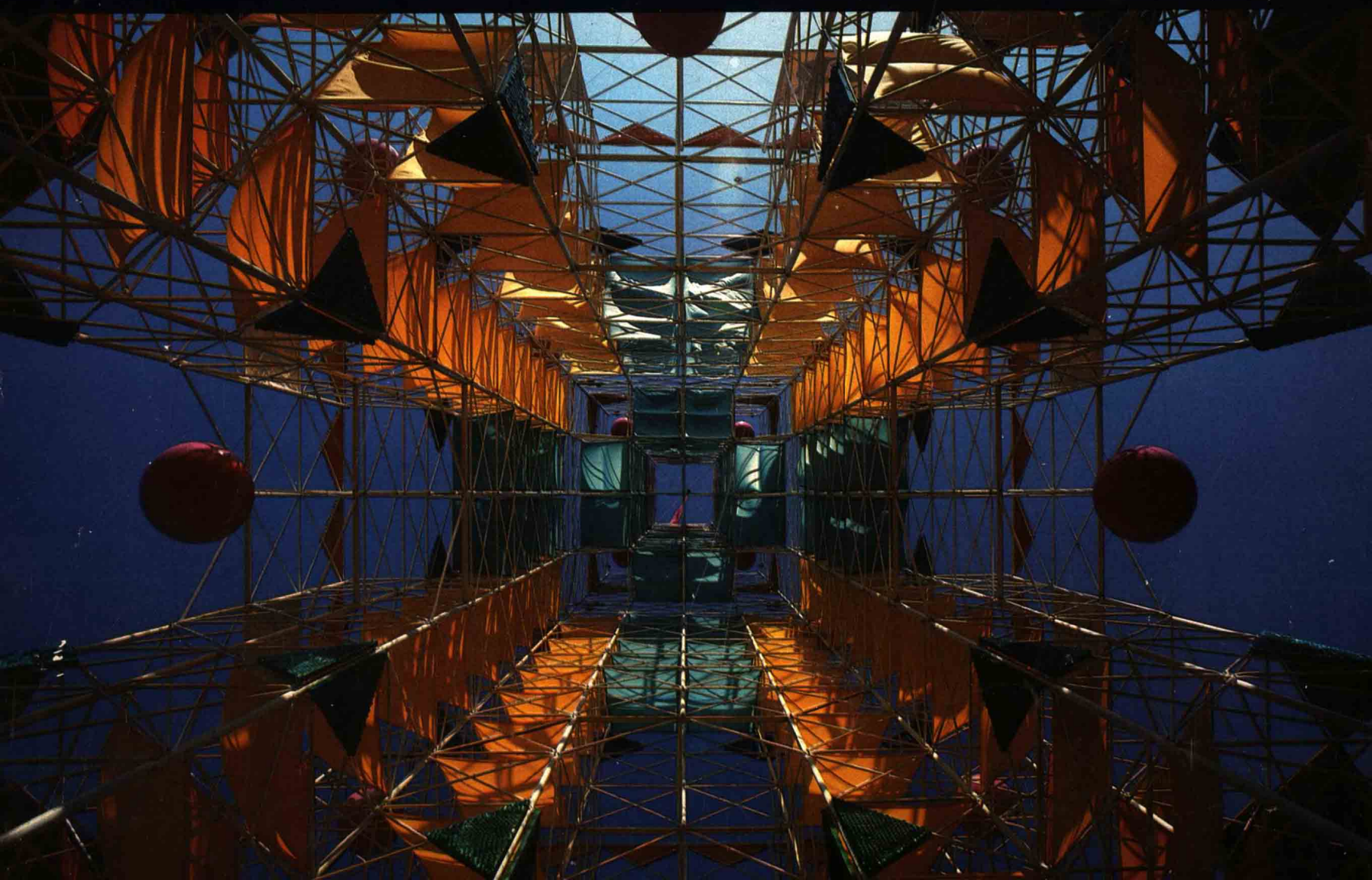
SECOND EDITION

Pascal

NELL DALE
CHIP WEEMS



Introduction to Pascal and Structured Design

NELL DALE
University of Texas at Austin

CHIP WEEMS
University of Massachusetts at Amherst

D. C. HEATH AND COMPANY
Lexington, Massachusetts Toronto

This book is dedicated to you,
and to all of our other students for whom it was begun and
without whom it would never have been completed.

Acquisitions Editor: Pam Kirshen
Developmental Editor: Lee Ripley
Production Editor: Marret McCorkle
Designer: Mark Fowler
Production Coordinator: Michael O'Dea
Photo Researcher: Martha Shethar
Text Permissions Editor: Margaret Roll

Trademark Acknowledgements: Turbo Pascal is a trademark of Borland International, Inc. Macintosh is a trademark of Apple Computer, Inc. VAX Pascal is a trademark of Digital Equipment Corporation. CDC is a trademark of Control Data Corporation. UCSD is a trademark of the Regents of the University of California. Pascal/M is a trademark of Sorcim. Pascal/MT+ and CP/M are trademarks of Digital Research. Pascal/Z is a trademark of Ithaca Intersystems.

Cover: 1984 Los Angeles Olympics Site Structure. Bill Gallery/Stock Boston

Copyright © 1987 by D. C. Heath and Company.
Previous edition copyright © 1983 by D. C. Heath and Company.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage or retrieval system, without permission in writing from the publisher.

Published simultaneously in Canada.

Printed in the United States of America.

International Standard Book Number: 0-669-09570-2 (paperback)

International Standard Book Number: 0-669-10399-3 (hardcover)

Library of Congress Catalog Card Number: 86-80506

Preface

> In the past there have been two distinct approaches used in introductory computer science texts. One approach focused on problem solving and algorithm design in the abstract, leaving the learning of a particular language to a supplemental manual or to a subsequent course. The second approach focused on the syntax of a particular programming language, and assumed that the problem-solving skills would be learned later through practice.
>
> We believe that neither approach is adequate. Problem solving is a skill that can and should be taught—but not in the abstract. Students must be exposed to the precision and detail required in actually implementing their algorithms in a real programming language.

The introduction to the Preface of the first edition of *Introduction to Pascal and Structured Design* (quoted above) proved to be prophetic as to the direction that computer science education would take. That edition came out in early 1983.

In 1983 the College Board published the description of an advanced placement course in computer science. In 1984 the ACM published the revised recommended curriculum for CS1, the first course in computer science. Both guidelines emphasize problem solving and algorithm design within the context of a block-structured language such as Pascal.

Since the first edition of this book has been widely accepted as a model for textbooks for CS1 and the first section of the AP exam in computer science, the temptation is to make only minimal changes for this second edition. We have resisted that temptation: to succumb would be to betray those students for whom the book was written.

The first edition of *Introduction to Pascal and Structured Design* was the first of a new wave of introductory textbooks. We trust this second edition will also make waves, for it is based on our vision of where computer science education is going: toward more testing, more abstraction, and more attention to the development of control structures and data structures.

Many topics considered advanced are introduced right from the beginning. Loop invariants are introduced as a way to design loops, not verify them. Designing

test data is included as an integral part of the programming process. Data types are defined as a set of values and the operations defined on those values. Parallel decomposition of problem and data structure is introduced with the first structured data types. Control and data abstraction are introduced early and encouraged throughout.

In response to your feedback, we have also included many more exercises and examples, earlier coverage of procedures, and more emphasis on interactive programming.

With all the changes, however, we have kept in mind the pedagogical philosophy that the best examples are those drawn from everyday experience. All problems and examples have been carefully chosen to require only high school algebra.

Organization

The order of presentation has been altered slightly to reflect our own changing view and the view of our colleagues who used the first edition.

Chapter 1 is still designed to create a comfortable rapport between the student and the subject. Most students now take their first course in an interactive programming environment. The discussion of the program entry, compilation, and execution process reflects this change with a shift in orientation toward timesharing systems and personal computers. Because it is still widely used in production environments, batch processing is also discussed.

By the end of Chapter 1 students should have a basic knowledge of what computers are, what programming is, and the mechanics of getting a program to run. The goal of Chapter 2 is to bring students to the point where they can design a simple program of their own. Because this involves so many fundamental concepts, we have chosen to divide the chapter into two parts. The first part introduces the bare minimum necessary to design and write a very simple program. The second part fleshes out the details of Pascal syntax for more complex expressions and output.

The top-down design methodology is a major focus of Chapter 3. Our discussion of the methodology builds on the problem-solving techniques that are introduced in Chapter 2 by providing a concrete framework in which to apply them. Chapter 3 also covers input from fundamental concepts to the finer points of style in writing prompting messages. Files other than Input and Output are introduced at this stage in order to allow instructors to assign programming problems that require the use of sample data files.

Both Chapters 2 and 3 contain discussions of procedures and functions, with an introduction to the basic concepts of subprogram calls, parameter passing, and subprogram libraries. Chapter 3 also relates subprograms to the principles of modular design that are used throughout the text.

Chapters 4 and 5 are devoted to the concepts of flow of control and the logical versus physical ordering of statements. In Chapter 4 we introduce selection with the IF-THEN-ELSE and IF-THEN statements. The concept of nested control structures is also developed in this chapter.

Chapter 5 is devoted to looping structures. As in the first edition, all of the structures are introduced using the syntax of the WHILE statement. Rather than

此为试读,需要完整PDF请访问: www.ertongbook.com

confuse the student with multiple syntactical structures, our approach is to teach the concepts of looping using only the WHILE. Students are first introduced to the basic loop control strategies and common looping operations. We then present a step-by-step process for designing loops using loop invariants.

Because there are so many new concepts associated with designing and writing user-defined procedures and functions, we have devoted three chapters to this topic. Chapter 6 covers flow of control in procedures, formal and actual parameters, interface design, local variables, and multiple calls to a procedure. Chapter 7 expands the discussion to include value parameters, nested scope, stubs and drivers, and more on interface design. Chapter 8 covers user-defined functions and briefly introduces recursion. Because of the numerical orientation of Chapter 8, we also take the opportunity to discuss the problems of representation and precision associated with Real numbers.

Chapter 9 represents a transition between the control structure orientation of the first half of the book and the abstract data type orientation of the second half. The chapter begins by introducing the first new data type since Chapter 3 (Sets) and ends by covering the remaining "ice cream and cake" control structures in Pascal (CASE, REPEAT, and FOR). Chapter 9 forms a natural ending point for the first quarter of a two-quarter introductory programming course.

In keeping with the increased emphasis on abstraction, simple data types are given a chapter all of their own, Chapter 10. The built-in simple data types are examined in terms of the set of values that variables or constants of that type can contain and the allowable operations on values of that type. Enumerated and subrange types are covered in detail. The functions Pred, Succ, and Ord are defined as Pascal implementations of the corresponding operation on scalar data types. Type compatibility is defined, and anonymous typing is strongly discouraged.

The array data type is introduced in Chapter 11. Arrays are the last big conceptual hurdle for the students to cross: A variable to access another variable? Three case studies and numerous small examples successfully make the jump. Three typical types of array processing are covered in the case studies: using only a portion of the array (subarray processing), using two or more arrays in parallel (parallel arrays), and using indices that have more meaning other than just representing a position (indexes with semantic content).

Chapter 12 represents a radical departure from the first edition. Algorithms that are commonly applied to lists of data are developed and coded as general-purpose Pascal procedures. Strings are described. A concluding Problem Solving in Action applies several of the procedures written in the first part of the chapter to strings to demonstrate the general applicability of the procedures.

Multidimensional arrays are discussed in Chapter 13; records are presented in Chapter 14 along with a lengthy discussion on how to choose an appropriate data structure. Data abstraction is demonstrated by creating an abstract data type Date and several useful operations on dates.

The remaining data types, files and pointers, are discussed in Chapter 15. Pointers are presented as a way to make programs more efficient. The use of pointers to create dynamic data structures is handled in a chapter by itself, Chapter 16. Linked lists in general and linked-list representations of stacks, queues, and binary trees are described briefly.

Chapter 17 deals with recursion. There is no consensus as to the best place to cover recursion. We personally feel that it is a topic that requires more maturity than many first semester students possess. We have included it, however, for two reasons: many instructors have requested it and there are those students for whom recursion seems natural. Although it is the last chapter, the examples are divided into two parts: those that require only simple data types and those that require structured data types. The first part is appropriate after Chapter 8. The second part contains examples from simple arrays to binary trees. These examples could be used singly after the appropriate chapter or as a unit after Chapter 16.

Additional Features

Problem Solving in Action Problem solving is demonstrated using case studies. A problem is presented followed by a discussion of how the problem might be solved by hand. An algorithm is developed using top-down design, and the algorithm is coded in Pascal. Sample test data and output are shown, followed by a discussion of what is involved in thoroughly testing the program.

Goals Goals for each chapter are listed at the beginning of the chapter. These goals are then tested in the end-of-chapter exercises.

Quick Checks At the end of each chapter there are questions to test the student's recall of major points keyed to the appropriate pages. The answers immediately follow in the body of the text.

Exam Preparation Exercises Thought questions to help students prepare for tests are presented. Answers to selected questions from each chapter are in the back of the book. Answers to the remaining questions are in the Instructor's Guide.

Preprogramming Exercises Questions that provide students with experience in writing Pascal code fragments or procedures are given in this section. This allows students to practice the syntactic constructs in each chapter without the overhead of writing a complete program. Solutions to selected questions from each chapter appear in the back of the book; the balance are given in the Instructor's Guide.

Programming Problems Specifications for problems from a wide range of disciplines are included. Students are required to write complete programs.

Supplements

Instructor's Guide Prepared by the authors, the Instructor's Guide includes teaching notes, answers to the balance of the exercises, a carefully worked out solution and discussion to one programming problem per chapter, and an example advanced placement exam question with a sample solution and the actual grading rubrics used by the AP exam graders.

Test Item File Prepared by Tom Parks, the Test Item File includes over 1200 possible test questions. It is available in Archive, a computerized test generator, for the IBM PC.

Compiler Supplements Supplementary booklets are available with compiler-specific information keyed to pages in the text. Three versions are available: MacIntosh™ Pascal, VAX Pascal™, and Turbo Pascal™.

In addition to the elements listed above, the programs in the text are available on disk in either Turbo format or ANSI format. A separate set of transparency masters is available to adopters of the text.

Acknowledgments

We would like to thank the many individuals who have helped us. We are indebted to the members of the faculties of the Computer Sciences Department at the University of Texas at Austin and of the Computer and Information Science Department at the University of Massachusetts at Amherst.

Among those in Austin, we would like to thank the following people: colleagues Jean Rogers and Joyce Brennan, who many times acted as a sounding board for new ideas; Carolyn Goldston, who never forgets birthdays; the Advance Placement CS readers who shared their ideas so willingly during those weeks at Rider College; and most especially Al Dale.

From among our colleagues in Amherst, we especially thank Caxton Foster, Alan Hanson, Steven Levitan, Edward Riseman, William Verts, and Beverly Woolf. Thanks also to Jeffrey Bonar of the Learning Research and Development Center at the University of Pittsburgh.

For their many helpful suggestions, we thank the lecturers, teaching assistants, consultants, and student proctors who run the courses for which this book was written, and the students themselves.

We would like to thank the following people who reviewed the manuscript: Randy Bartell, Syracuse University; James Case, Hiram College; Thomas Copeland, Middlebury College; John A. Koch, Wilkes College; Russell B. Lee, Allan Hancock College; Teck-Kah Lim, Drexel University; Ken Loach, State University of New York at Plattsburgh; Joseph Mayne, Loyola University; Randall Jay Molmen, University of North Dakota; Jean Rogers, University of Texas at Austin; Harbans Sathi, University of Southern Colorado; Mary Lou Soffa, University of Pittsburgh; Richard St. Andre, Central Michigan University; Bernard Taheny, Chicago Public Schools; Tim Thurman, University of Kansas; Darrell R. Turnidge, Kent State University; Frank T. Vanecek, Norwich University; Greg Wetzel, University of Kansas; D. Franklin Wright, Cerritos College.

For this impressive list of reviewers, as well as her tremendous support, we must thank our editor, Pam Kirshen. To all the others at D. C. Heath who contributed so much, especially Lee Ripley, Susan Gleason, Ruth Thompson, and Marret McCorkle, we are indeed grateful.

Special thanks go to David Orshalick for his input at the early stages of the development of this edition.

Anyone who has ever written a book—or is related to anyone who has—knows the amount of time involved in such a project. To our families who learned this first hand, all we can say is: "To Sarah, Susy, June, Judy, Bobby, Phil, Carol, and Lisa, thanks for your tremendous support and indulgence."

N.D.
C.W.

GOALS

- *To gain an understanding of what a computer program is.*
- *To be able to list the basic steps involved in writing a program.*
- *To be able to define what an algorithm is.*
- *To learn what the major parts of a computer are and how they work together.*
- *To learn the difference between interactive and batch processing.*
- *To learn the difference between hardware and software.*
- *To know what a programming language is.*
- *To understand the compilation and execution processes.*
- *To learn some of the history and features associated with the Pascal programming language.*
- *To learn the steps involved in entering a program into the computer and getting it to run correctly.*

Contents

1 **OVERVIEW OF PROGRAMMING** ____ 1

Why Programming? 1
What Is Programming? 3
Data Representation 7
What Is a Computer? 8
Personal Computers and Mainframes 13
What Is a Programming Language? 16
What Is Pascal? 21
Program Entry, Correction, and Execution 24
Interactive Program Entry 25
Batch Program Entry 31
SUMMARY 32 • *QUICK CHECK* 33 • *EXAM PREPARATION EXERCISES* 34

2 **PROBLEM SOLVING, SYNTAX/SEMANTICS, AND PASCAL PROGRAMS** ____ 36

Part 1 ALGORITHMS, DATA, AND PROGRAM CONSTRUCTION 36
Problem-Solving Process 37
Ask Questions 38
Look for Things That Are Familiar 38
Divide and Conquer 38
Solve by Analogy 39

此为试读,需要完整PDF请访问: www.ertongbook.com

The Building Block Approach 40
Means-Ends Analysis 40
Mental Blocks: The Fear of Starting 42
Syntax/Semantics 44
Syntax Diagrams 44
Identifiers 45
Data Types 47
Data Storage 49
Assignment 53
Output 57
Program Construction 59
Compound Statements 61
Blocks 61
Program Formatting 62
SUMMARY 65 • *QUICK CHECK* 66

Part 2 MORE OUTPUT MORE EXPRESSIONS 69
The Writeln Statement 70
The Appearance of Output 72
Formatting Output 73
Formatting Integer and Character Output 74
Program ForMom, Written with a Procedure 78
Formatting Real Output 79
More Expressions 80
Precedence Rules 80
Functions 81
Programs in Memory 86
SUMMARY 86 • *QUICK CHECK* 88 • *EXAM PREPARATION EXERCISES* 89
PREPROGRAMMING EXERCISES 92 • *PROGRAMMING PROBLEMS* 93

3 *INPUT AND DESIGN METHODOLOGY* 94
Getting Data into Programs 95
Read 96
Readln 98
The Reading Marker and <eoln> 99
Reading Character Data 100
More About Procedures and Parameters 103

Interactive Input/Output 103
Batch Input/Output 105
Files Other Than Input and Output 106
Using Files 107
Listing File Names in the Program Heading 107
Declaring Files in the VAR Section 108
Preparing Files with Reset or Rewrite 108
Specifying Files in Read, Readln, Write, and Writeln 110
The Impact of Data Formats on Program Design 112
Top-Down Design 112
Modules 113
Methodology 117
Documentation 119
Testing and Debugging 126
SUMMARY 128 • *QUICK CHECK* 129 • *EXAM PREPARATION EXERCISES* 129 • *PREPROGRAMMING EXERCISES* 131 • *PROGRAMMING PROBLEMS* 132

4 **SELECTION** 134

Conditions and Boolean Expressions 136
Boolean Expressions 136
Precedence of Operators 140
Writing Boolean Expressions 141
Relational Operators with Real and Boolean Data Types 142
The Boolean Function Odd 142
Selection Control Structures 143
Flow of Control 143
Selection 143
The IF Statement 143
The IF-THEN-ELSE Form of IF Statement 144
Compound Statements 145
The IF-THEN Form of IF Statement 147
Nested IF Statements 151
Testing and Debugging 161
Testing and Debugging Hints 163

SUMMARY 165 • QUICK CHECK 165 • EXAM PREPARATION EXERCISES 166 • PREPROGRAMMING EXERCISES 168 • PROGRAMMING PROBLEMS 170

5 LOOPING 172

WHILE Statement 173
Loops Using the WHILE Statement 175
Count-Controlled Loops 176
Event-Controlled Loops 177
Looping Subtasks 183
How to Design Loops 186
Determining the Loop Invariant 188
Designing the Flow of Control for a Loop 188
Designing the Process Within the Loop 190
Nested Logic 206
Designing Nested Loops 209
Printing Headings and Columns 210
Testing and Debugging 217
Testing and Debugging Hints 218

SUMMARY 219 • QUICK CHECK 220 • EXAM PREPARATION EXERCISES 221 • PREPROGRAMMING EXERCISES 223 • PROGRAMMING PROBLEMS 224

6 PROCEDURES 226

Top-Down Structured Design with Procedures 227
An Overview of User-Defined Procedures 234
Flow of Control in Procedure Calls 234
When to Use Procedures 234
Procedures and Blocks 235
Parameters 236
An Analogy 236
Procedure Declarations 238
Procedure Call (Invocation) 238
Naming Procedures 239
Parameters 239
Local Variables 242

Multiple Calls to the Same Procedure 243
Testing and Debugging 249
Testing and Debugging Hints 250

SUMMARY 250 • *QUICK CHECK* 251 • *EXAM PREPARATION EXERCISES* 252 • *PREPROGRAMMING EXERCISES* 254 • *PROGRAMMING PROBLEMS* 256

7 ***VALUE PARAMETERS AND NESTED SCOPE*** 258

VAR/Value Parameters 259
Value Parameter Semantics 260
Interface Design 260
Value Parameter Syntax 262
Local versus Global Declarations 270
Scope Rules 272
Side Effects 277
Global Constants 280
Designing Programs with Nesting 280
Testing and Debugging 282
Stubs and Drivers 287
Testing and Debugging Hints 291

SUMMARY 292 • *QUICK CHECK* 293 • *EXAM PREPARATION EXERCISES* 293 • *PREPROGRAMMING EXERCISES* 297 • *PROGRAMMING PROBLEMS* 299

8 ***FUNCTIONS, PRECISION, AND RECURSION*** 302

Functions 303
Boolean Functions 306
Function Interface Design and Side Effects 307
When to Use Functions 308
More on Real Numbers 316
Representation of Real Numbers 316
Arithmetic with Real Numbers 319
How Pascal Implements Real Numbers 320
Practical Implications of Limited Precision 322
Recursion 323
Testing and Debugging 326
Testing and Debugging Hints 326

SUMMARY 327 • QUICK CHECK 328 • EXAM PREPARATION EXERCISES 328 • PREPROGRAMMING EXERCISES 330 • PROGRAMMING PROBLEMS 330

9 SETS AND ADDITIONAL CONTROL STRUCTURES 332

Sets 333
Additional Control Structures 343
REPEAT Statement 343
FOR Statement 346
Guidelines for Choosing a Looping Statement 348
CASE Statement 349
Testing and Debugging 356
Testing and Debugging Hints 356

SUMMARY 357 • QUICK CHECK 358 • EXAM PREPARATION EXERCISES 359 • PREPROGRAMMING EXERCISES 360 • PROGRAMMING PROBLEMS 362

10 SIMPLE DATA TYPES 364

Data Types 365
Ord, Pred, and Succ Functions 367
Chr Function 369
User-Defined Scalar Data Types 370
Enumerated Data Types 371
Subrange Types 383
Anonymous and Named Data Types 385
Type Compatibility 386
Sets and Additional Control Structures Revisited 389
Testing and Debugging 397
Testing and Debugging Hints 401

SUMMARY 402 • QUICK CHECK 402 • EXAM PREPARATION EXERCISES 403 • PREPROGRAMMING EXERCISES 404 • PROGRAMMING PROBLEMS 404

11 ONE-DIMENSIONAL ARRAYS 406

Structured Data Types 407
One-Dimensional Arrays 414
Defining Arrays 414
Accessing Individual Components 416

Examples of Defining and Accessing Arrays 416
Processing an Array 421
Using Arrays in Programs 423
Subarray Processing 423
Parallel Arrays 423
Indices with Semantic Content 424
Special Note on Passing Arrays as Parameters 438
Testing and Debugging 438
Testing and Debugging Hints 439
SUMMARY 440 • QUICK CHECK 440 • EXAM PREPARATION EXERCISES 442 • PREPROGRAMMING EXERCISES 443 • PROGRAMMING PROBLEMS 444

12 APPLIED ARRAYS 448

Algorithms on Lists 449
Sequential Search in an Unordered List 450
Sorting 453
Sequential Search in a Sorted List 456
Inserting into an Ordered List 458
Binary Search in an Ordered List 461
Working with Words 466
Testing and Debugging 485
Testing and Debugging Hints 487
SUMMARY 487 • QUICK CHECK 488 • EXAM PREPARATION EXERCISES 489 • PREPROGRAMMING EXERCISES 491 • PROGRAMMING PROBLEMS 492

13 MULTIDIMENSIONAL ARRAYS 494

Two-Dimensional Arrays 496
More on Array Processing 508
Initialize the Table 510
Sum the Rows 511
Sum the Columns 512
Print the Table 513
Another Way of Defining Two-Dimensional Arrays 525
Multidimensional Arrays 527

此为试读,需要完整PDF请访问: www.ertongbook.com