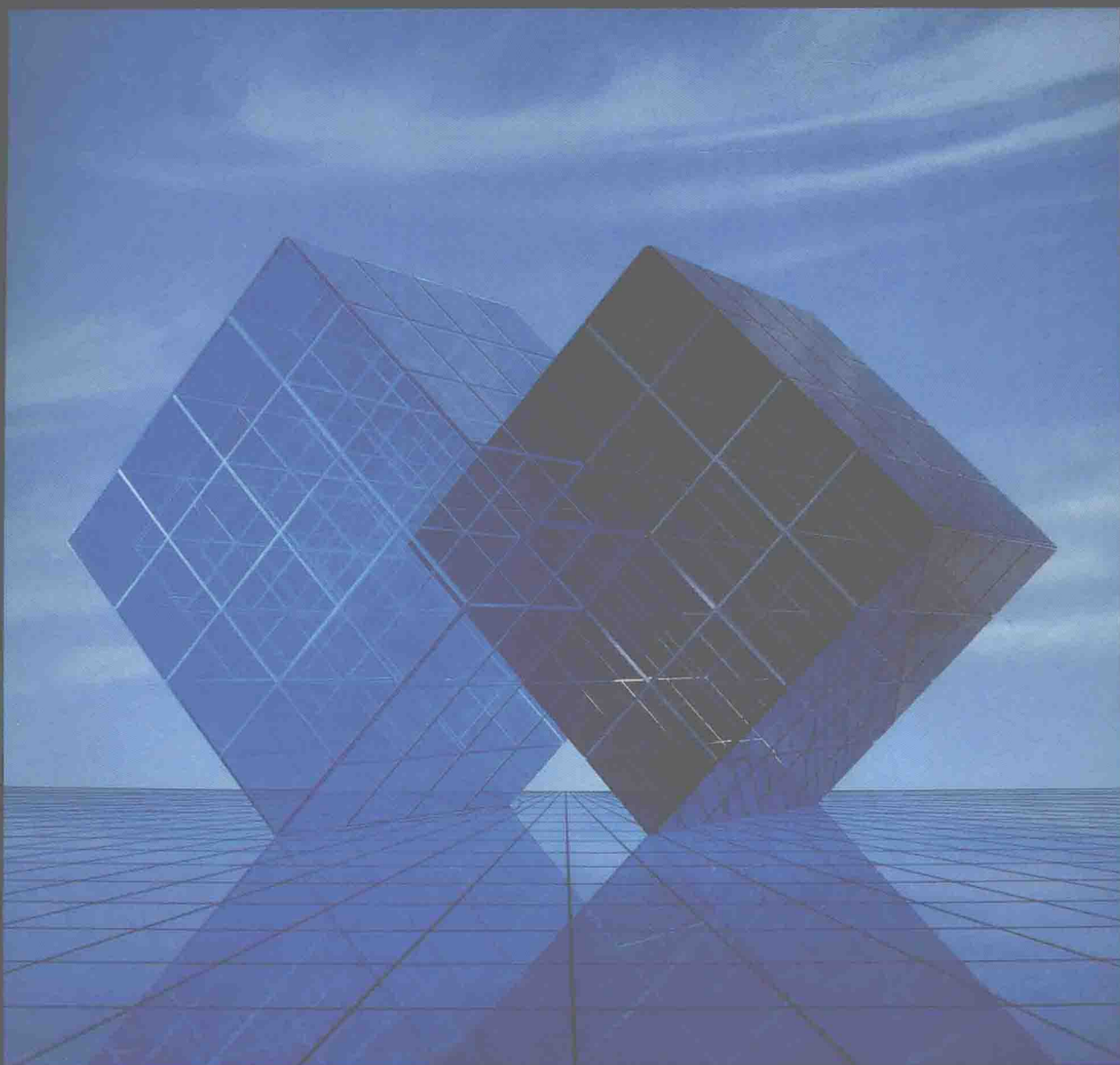


# FUNDAMENTAL PROGRAMMING WITH PASCAL

---



---

J. DENBIGH STARKEY  
ROCKFORD J. ROSS

# **Fundamental Programming With Pascal**

**J. Denbigh Starkey**  
Washington State University

**Rockford J. Ross**  
Montana State University

**West Publishing Company**  
St. Paul • New York • Los Angeles • San Francisco

Cover photography by Michel Tcherevkoff for Alleghany.

COPYRIGHT © 1984 By WEST PUBLISHING CO.

50 West Kellogg Boulevard

P.O. Box 43526

St. Paul, Minnesota 55165

All rights reserved

Printed in the United States of America

Library of Congress Cataloging in Publication Data

Starkey, J. Denbigh.

Fundamental programming with Pascal.

Bibliography: p.

Includes index.

1. PASCAL (Computer program language) I. Ross,  
Rockford. II. Title.

QA76.73.P2573 1984

001.64'24

84-2244

ISBN 0-314-77806-3

# **Fundamental Programming With Pascal**

To Susan  
and  
To Heidi, Jason, and Jennifer

# Preface

Our purpose in writing this book was to present an introduction to programming in true textbook form. What we wanted was a textbook that could be used by students and instructors alike with the confidence that the issues fundamental to programming were covered in a thorough and integrated fashion. We wanted to ensure that students were not left dangling by incomplete discussions or topics devoid of application, and that instructors would not be forced to refer to other sources to fill in gaps left in the book. What we did *not* want was yet another Pascal programming language manual (many fine Pascal manuals already exist) or a book that only discussed the fundamental issues of programming in isolated sections without integrating these concepts into the everyday development of programs. Learning the details of a programming language like Pascal is relatively easy; the challenging part is learning how to program.

In striving towards our goals we have held to one basic tenet: students learn best by example. Each new programming concept introduced is discussed as it is needed within the framework of a complete top-down development of a program. Related issues, such as program efficiency and correctness, are integrated within this discussion rather than being relegated to isolated sections. Thus, this is a large book, not because it encompasses so much new material, but because of the novel and thorough presentation of the fundamental topics.

## PHILOSOPHY

In determining the content of the book we had a definite pedagogical model in mind. This model stemmed from the startling observation that when students were asked to design a substantial program independently, many would return with a program consisting primarily of one or two large routines with little modularity — programs that were difficult to read and difficult to modify. Furthermore, when asked simple questions about the efficiency or correctness of their programs, the students were often at a loss for answers. This was true in spite of our efforts at teaching structured design, correctness, and efficiency of programs. What had gone wrong? The answer was surprisingly simple: we weren't practicing what we preached. Traditional textbooks used in the introductory courses either did not cover these topics well or they covered them in isolated sections of the text. Furthermore, procedures and functions were normally introduced late in these textbooks, almost as an afterthought, as the "right way" to program. Students were mistakenly led to believe that procedures and functions were difficult topics of more bother than they were worth; it's no wonder that they were avoiding their use later. In designing our book, then, our philosophy was to introduce programming in a way that would reinforce proper programming style and habits from the start. We do this as follows:

**(1) Case Studies.** The central pedagogical tool we use is the case study. These are programming problems for which complete, working programs are designed in a top-down, structured fashion as new programming concepts are introduced. The problem of exploring new concepts in isolation from practical experience is thus avoided. In all there are 51 complete case studies in the book.

**(2) Use of a Pseudolanguage.** The solutions to the case studies are developed in a structured, top-down fashion in a simple pseudolanguage. This allows us to concentrate on programming rather than the distracting details of Pascal as the programs are developed. Students should learn that program development in a pseudolanguage is a completely separate process (now widely practiced in industry) from the implementation of the resulting program in some particular programming language (in this case Pascal). Each of the programs we design in the pseudolanguage is translated into a complete, working Pascal program in a later section where the new details of the Pascal language can be discussed separately from the problems involved in the program design.

**(3) Immediate Introduction of Procedures and Functions.** From the first case study on we teach that programs are collections of short, well-defined procedures and functions, which are organized and called from an initial procedure (main program). The crucial concepts of procedures, functions, parameters, and modular program design are thus ingrained into the habits and practice of students from the beginning. Students learn these topics without problem, and their later programming practices are greatly enhanced as a result.

**(4) Inclusion of Program Correctness.** As part of each case study we include an integrated discussion of program correctness. This starts out quite simply with the early case studies but eventually includes the notions of a program

walkthrough, semi-formal verification steps (particularly for loops), program testing, choosing proper test data, robustness, and debugging techniques. Students receive a practical knowledge of the concepts of program verification and are provided a sound basis for advanced courses on the topic.

**(5) Integrated Discussion of Program Efficiency.** The execution time efficiency (time complexity) and storage space requirements (space complexity) of the programs are discussed for each relevant case study. Time complexity is determined by doing a count of the number of statements executed, and space complexity is determined by counting the number of storage cells used. These simple, intuitive approaches are accurate and practical. Students continuing on in computer science will have a basis for advanced study of these topics, while those terminating after this class will understand practical methods for determining program efficiency.

## BOOK ORGANIZATION

All chapters except the first follow a specific format designed to implement our philosophy. Each has four major sections, **Getting Acquainted**, **In Retrospect**, **The Challenge**, and **Pascal Implementation**. In the Getting Acquainted section, simple case studies introducing the new programming concepts of the chapter are studied. All of these case studies should be covered because the procedures and functions developed there are often used in later case studies. In Retrospect summarizes these new concepts and provides a place to turn to for review. The Challenge presents more challenging case studies involving the new concepts of the chapter. The Pascal Implementation section mirrors the previous sections exactly in translating the pseudolanguage programs of the case studies into Pascal. All Pascal programs have been written to conform to the ISO standard. Appendix A, **Pascal Reference**, provides a concise reference manual for ISO Pascal. To help acquaint students with this language reference appendix, syntax diagrams from the appendix should be presented in class as new Pascal statements are introduced. Eight groups of exercises are integrated into each chapter, and answers to some of these are found in appendix B, **Answers to Selected Exercises**.

The first chapter of the text is different from the other chapters; it describes a model computer and the simple operations that a computer can perform, providing the motivation for the rest of the book by answering the question, "Why must we write programs?". It was carefully written so that students could read it on their own during the first week of class as the instructor tended to other matters (such as describing how to use the computer). A complete, simple Pascal program is given at the end of the exercises, which the students can type in and run as their first assignment to help acquaint them with their computer terminal and text editor.

We hope that you will find this book as easy and pleasant to use as we have. However, you will probably find parts that you dislike or disagree with. We would be delighted to receive any comments you may have, and corrections will be gratefully accepted and included in future printings or editions.

## IN GRATITUDE

Those to whom we are most indebted for the form and content of this book are the thousands of students in the introductory programming course at



Washington State University over the past two years. Their questions, comments, sharp-eyed ability to catch errors, and enthusiasm for the material kept us going. Also to be thanked are the many reviewers who helped us through various stages of the book:

Gabriel Barta (University of New Hampshire)	Rachelle Heller (University of Maryland)
Rodney M. Bates (Kansas State University)	Leon Levine (University of California, Los Angeles)
Leland L. Beck (San Diego State University)	Gene Mahalko (University of North Dakota)
Don Cartlidge (New Mexico State University)	Lawrence H. Miller (University of California, Los Angeles)
Cecelia R. Daly (University of Nebraska)	Ralph Moore (Modesto Junior College, California)
Nancy Duffrin (SUNY at Stony Brook)	Keith R. Pierce (University of Minnesota)
Arthur C. Fleck (University of Iowa)	Alan L. Schwartz (University of Missouri, St. Louis)
Tamar E. Granor (University of Pennsylvania)	Robert F. Simmons (University of Texas, Austin)
James L. Hein (Portland State University)	Stephen F. Weiss (University of North Carolina, Chapel Hill)

Their careful evaluations and widespread support for the book were indeed helpful. The many instructors who labored in front of the classes with rough drafts of the book receive our appreciation, too.

Although many helped us along the way, there are others whose help was particularly important. Roger Hirsch first interested us in writing the book, without his encouragement we might never have started. Shirley Farmer was unflagging in her typing of the manuscript (even when her eyes said "Oh no, not chapter 4 again!") We also thank Mike Langston, whose enthusiasm for the book and personal friendship as a colleague did not temper his constructive criticism of our efforts. Then there are all those at West Publishing Company, who probably never saw so many deadlines come and go before and yet continued to provide the necessary support to keep the project moving. Editor Pete Marshall, production editor Deanna Quinn, and marketing coordinator Reneé Grevious worked so intensely with us that they now seem more like friends than business associates. We also thank Pamela McMurry for carefully copyediting our manuscript; she not only read for syntactic errors but for a true understanding of the material.

Then there are the few without whose support the project definitely would have been doomed: our families. To those who missed us on family occasions, holidays, and weekends, yet continued to stand by as the project seemed to extend indefinitely — well, what can we say? Without you there would be no book.

**J. Denbigh Starkey  
Rockford J. Ross**

# To the Student

This book has been designed with you in mind. We have given numerous examples of all important programming concepts and provided exercises to reinforce your learning. If you study this material carefully you will be well prepared for advanced courses in computer science; if this is the only course that you plan to take, you will have a sound understanding of the programming process to apply in later life. For example, it may well be that the most useful things that a future engineer learns from this book are program correctness and efficiency if he or she is later involved in projects where the successful design of a program by members of a team is crucial, or where the speed of a particular software component of a system is important. Similarly, business students may later find that they are responsible for decisions about the purchase or use of programs, and a practical, working knowledge of the concepts of program design, efficiency, and correctness may be far more important than actual programming skills. In short, these topics are of concern not only to computer professionals but to all who will be involved with computers in the future.

To use this book most effectively, you should read the chapters in the order given. The first chapter answers the question, "Why must computers be programmed?" You can read it as you are becoming used to the actual computer you are using; this chapter contains no programming assignments. In subsequent chapters you should study each of the case studies in the Getting Acquainted sections in succession. The order is important because later case studies build on earlier ones. After you understand a case study you should turn to the corresponding Pascal section to see how the program developed in the case study is translated into Pascal. This process of learning the design of a program and then turning to the Pascal section to see how the program is actually written in Pascal is one you will repeat often. The In Retrospect section of each chapter is especially for you. You should read it carefully and then use it as a reference whenever you need to review a particular topic. The Challenge section contains advanced case studies exploring the new topics of the chapter. Later introductory case studies do not depend on previous Challenge case studies, but the Challenge case studies give you a more intimate look at the programming process. You should also learn to use appendix A, Pascal Reference, as quickly as possible. It gives a detailed but concise description of Pascal, and it should be used whenever you want to review features of the Pascal programming language.

Whether you are a computer science student or a student from another discipline we have designed this book to be useful to you now and later as a reference. And one warning: if you have learned a programming language previously on your own, try to forget what you learned. We have seen many sad cases of students coming in with previous programming experience who start well but end up doing poorly because they never shook their previously learned bad habits! Using this book you will not only learn Pascal, you will also learn to be a good programmer.

# Contents

PREFACE    xix

TO THE STUDENT    xxiii

## CHAPTER 1 GETTING READY    1

1.1 BACKGROUND    2

1.2 THE MODEL COMPUTER    3

The Processor    4/The Store    8/The Processor — Comparison  
Unit    19/The Input and Output Devices    30/Summary of the  
Model Computer    38

1.3 THE PROGRAMMING PROCESS    38

Programs    38/Functions and Procedures    39/The Model  
Programming Language    42

1.4 EXERCISES    44

Review Exercises    44

1.5 PROGRAM IMPLEMENTATION IN PASCAL    46

The VAR Statement and Pascal Variables    46/Assignment Statements  
in Pascal    49/Inputting Values in Pascal — The READ  
Statement    55/Outputting Values in Pascal — The WRITELN  
Statement    56/An Example Pascal Procedure    56

1.6 PASCAL EXERCISES    57

## CHAPTER 2 SIMPLE PROGRAMS    61

2.1 GETTING ACQUAINTED    62

Case Study 2.1 Compute the Average of Three Numbers    62

*First Solution — A Simple Initial Procedure    62/Second  
Solution — Use a Simple Function    64/Correctness    69/  
Summary — Case Study 2.1    74*

Case Study 2.2 Area of a Rectangle    75

*The Solution — A Function with In Only Parameters    75/  
Correctness    77/Summary — Case Study 2.2    79*

Case Study 2.3 Swap Two Values    79

*The Solution — Procedure Swap With In Out Parameters    79/  
Correctness    83/Summary — Case Study 2.3    85*

2.2 IN RETROSPECT 87  
Top-Down Program Design 87/Functions and Procedures 88/  
Program Structure 90/Parameters 91/Local Variables 101/  
Program Verification 101/Batch vs. Interactive Programming 102

2.3 WARMUP EXERCISES 103  
For Review 103/A Deeper Look 104/To Program 106

2.4 THE CHALLENGE 107  
Case Study 2.4 Computing Simple Interest 107  
    *The Solution — A Function to Compute Simple Interest* 107/  
    *Summary — Case Study 2.4* 109  
Case Study 2.5 Computing Compound Interest 109  
    *The Solution — A Function to Compute Compound Interest* 109/  
    *Summary — Case Study 2.5* 110  
Case Study 2.6 Compute the Area of a Trapezoid 111  
    *The Solution — Use Function Rectangle of Case Study 2.2* 111/  
    *Improved Solution — Computing the Area of a Trapezoid*  
    *Directly* 113/*Summary — Case Study 2.6* 113

2.5 WORKING OUT 113

2.6 PASCAL IMPLEMENTATION 115  
Getting Acquainted With Pascal 116  
Case Study 2.1 in Pascal 116  
Case Study 2.2 in Pascal 120  
Case Study 2.3 in Pascal 122  
Pascal in Retrospect 123  
    *Pascal Program Structure* 124/*Pascal Functions* 124/*Pascal*  
    *Procedures* 126/*Pascal Parameters* 127/*Integer and Real*  
    *Numbers in Pascal* 127/*Testing Pascal Programs* 129/*Pascal*  
    *Summary* 129  
Warming Up to Pascal 129  
    *For Review* 129/*A Deeper Look* 130/*To Program* 130  
The Challenge in Pascal 131  
Case Study 2.4 in Pascal 131  
Case Study 2.5 in Pascal 132  
Case Study 2.6 in Pascal 133  
Working Out in Pascal 134

**CHAPTER 3 MAKING DECISIONS 137**

3.1 GETTING ACQUAINTED 138  
Case Study 3.1 Find the Larger of Two Values 138  
    *Correctness* 139/*Summary — Case Study 3.1* 140  
Case Study 3.2 Compute the Largest of Three Values 141  
    *First Solution — Nested If-Then-Else* 141/*Correctness* 143/  
    *Second Solution — Use of a Local Variable* 145/*Third*  
    *Solution — Use Compound Conditional Expression* 145/*Fourth*  
    *Solution — Use Function Maximum* 147/*Summary — Case*  
    *Study 3.2* 149  
Case Study 3.3 Sort Three Integers 149  
    *The Solution — A Sorting Procedure* 149/*Correctness* 151/  
    *Summary — Case Study 3.3* 152

Case Study 3.4	Assigning Student Grades	152
	<i>First Solution — Nested If-Then-Else</i>	153/ <i>Improved Solution — If Statement with Elseif Clauses</i>
		154/ <i>Correctness</i>
	<i>Summary — Case Study 3.4</i>	156
Case Study 3.5	Compute Ticket Costs	156
	<i>First Solution — If with Elseif Clauses</i>	157/ <i>Improved Solution — Case Statement</i>
		157/ <i>Correctness</i>
	<i>Summary — Case Study 3.5</i>	160
3.2	IN RETROSPECT	160
	The If Statement	161/ <i>Case Statement</i>
		164/ <i>Program Correctness</i>
		166
3.3	WARMUP EXERCISES	168
	For Review	168/ <i>A Deeper Look</i>
		170/ <i>To Program</i>
		171
3.4	THE CHALLENGE	172
	Case Study 3.6	Factorial
		172
	<i>The Solution — A Recursive Function</i>	174/ <i>Correctness</i>
		175/ <i>Summary — Case Study 3.6</i>
		186
	Case Study 3.7	Fibonacci Numbers
		187
	<i>The Solution — A Recursive Function</i>	187/ <i>Summary — Case Study 3.7</i>
		189
	Case Study 3.8	Quadratic Equations
		190
	<i>The Solution — Two Procedures</i>	190/ <i>Correctness</i>
		195/ <i>Summary — Case Study 3.8</i>
		196
3.5	WORKING OUT	197
3.6	PASCAL IMPLEMENTATION	198
	Getting Acquainted With Pascal	198
	Case Study 3.1 in Pascal	199
	Case Study 3.2 in Pascal	200
	Case Study 3.3 in Pascal	203
	Case Study 3.4 in Pascal	204
	Case Study 3.5 in Pascal	205
	Pascal in Retrospect	207
	<i>Compound Statements</i>	207/ <i>If-Then and If-Then Else</i>
		207/ <i>Implementation of Elseif Clauses in Pascal</i>
		210/ <i>The Pascal CASE Statement</i>
		213/ <i>Controlling the Printing of Values in a WRITELN Statement</i>
		216/ <i>Assertions in Pascal</i>
		220/ <i>Portability</i>
		221
	Warming Up to Pascal	221
	<i>For Review</i>	221/ <i>A Deeper Look</i>
		223/ <i>To Program</i>
		224
	The Challenge in Pascal	225
	Case Study 3.6 in Pascal	225
	Case Study 3.7 in Pascal	225
	Case Study 3.8 in Pascal	226
	Working Out in Pascal	227

## CHAPTER 4 ITERATION 229

4.1	GETTING ACQUAINTED	230
	Case Study 4.1	Sum a Series of Integers
		230
	<i>The Solution — A While Loop</i>	230/ <i>Method 1 — Header Value</i>
		231/ <i>Correctness</i>
		234/ <i>Method 2 — Trailer Value</i>
		237/

Correctness	239/Method 3 — The Moredata Condition	239/
Correctness	241/Efficiency	241/Summary — Case
Study 4.1	243	
Case Study 4.2	Sum the First n Even Integers	244
The Solution — An Indexing Loop	244/Alternate Solution — An	
Automatic Indexing Loop	248/Correctness	250/
Efficiency	251/Summary — Case Study 4.2	252
Case Study 4.3	Computing Average Scores	253
The Solution — While Moredata and Automatic Indexing		
Loops	253/Correctness	256/Efficiency
256/Summary —		
Case Study 4.3	257	
Case Study 4.4	Accumulating Interest	257
The Solution — The General While Loop	257/An Alternative	
Solution — The Until Loop	260/Correctness	262/
Efficiency	262/Summary — Case Study 4.4	264
4.2	IN RETROSPECT	264
Loops	264/Common Loop Operations	270/Program
Efficiency	272/Determining Loop Correctness	287
4.3	WARMUP EXERCISES	302
For Review	302/A Deeper Look	303/To Program
307		
4.4	THE CHALLENGE	307
Case Study 4.5	Factorial Revisited	307
The Solution — A Loop	308/Correctness	309/
Efficiency	310/Summary — Case Study 4.5	310
Case Study 4.6	Fibonacci Revisited	311
The Solution — A Loop	311/Correctness	313/
Efficiency	313/Summary — Case Study 4.6	313
4.5	WORKING OUT	314
4.6	PASCAL IMPLEMENTATION	316
Getting Acquainted With Pascal	316	
Case Study 4.1 in Pascal	316	
Case Study 4.2 in Pascal	319	
Case Study 4.3 in Pascal	320	
Case Study 4.4 in Pascal	324	
Pascal in Retrospect	326	
The Pascal WHILE Loop	326/The Pascal REPEAT-UNTIL	
Loop	328/The Pascal FOR Loop	329/More Pleasing
Output	330/WRITE and WRITELN	330/READ and
READLN	333/Function and Procedure Nesting	337/The
FORWARD Statement	341/Global Variables	343
Warming Up to Pascal	347	
For Review	347/A Deeper Look	348/To Program
349		
The Challenge in Pascal	350	
Case Study 4.5 in Pascal	350	
Case Study 4.6 in Pascal	351	
Working Out in Pascal	351	

## CHAPTER 5 MAINTAINING SIMPLE LISTS: ARRAYS OF ONE DIMENSION 353

### 5.1 GETTING ACQUAINTED 354

- Case Study 5.1 Printing a List Forwards and Backwards 354
  - The Solution — 1-D Arrays* 355/*Inputting Values into a 1-D Array by the Header Method* 356/*Printing a 1-D Array in Reverse Order* 359/*Correctness* 360/*Inputting Values into a 1-D Array by the Moredata Method* 365/*Inputting Values into a 1-D Array by the Trailer Method* 368/*Efficiency* 369/*Summary — Case Study 5.1* 370
- Case Study 5.2 Computing Temperature Statistics 371
  - The Solution — Arrays with Negative Bounds* 371/*Correctness* 373/*Efficiency* 373/*Summary — Case Study 5.2* 374
- Case Study 5.3 Computing Test Statistics 374
  - The Solution — Statistics on 1-D Arrays* 375/*Correctness* 380/*Efficiency* 382/*Summary — Case Study 5.3* 382

### 5.2 IN RETROSPECT 383

- Simple Lists 383/*The 1-D Array* 383

### 5.3 WARMUP EXERCISES 389

- For Review 389/*A Deeper Look* 389/*To Program* 390

### 5.4 THE CHALLENGE 391

- Case Study 5.4 Computing the Dot Product of Two Vectors 391
  - The Solution — 1-D Real Arrays* 391/*Correctness* 393/*Efficiency* 393/*Summary — Case Study 5.4* 393
- Case Study 5.5 Searching an Unordered List 393
  - The Solution — Sequential Search* 394/*Correctness* 397/*Efficiency* 398/*Summary — Case Study 5.6* 398
- Case Study 5.6 Searching an Ordered List 399
  - First Solution — Sequential Search* 399/*Second Solution — Binary Search* 402/*Correctness* 404/*Efficiency* 406/*Summary — Case Study 5.6* 407
- Case Study 5.7 Sorting a List of Values 407
  - The Solution — Insertion Sort* 408/*Correctness* 414/*Efficiency* 414/*Summary — Case Study 5.7* 416
- Case Study 5.8 Sorting a List of Values Faster 416
  - The Solution — Quicksort* 417/*Correctness* 428/*Efficiency* 433/*Summary — Case Study 5.8* 436

### 5.5 WORKING OUT 436

### 5.6 PASCAL IMPLEMENTATION 438

- Getting Acquainted With Pascal 438
- Case Study 5.1 in Pascal 438
- Case Study 5.2 in Pascal 440
- Case Study 5.3 in Pascal 442
- Pascal in Retrospect 444
  - Direct 1-D Array Declarations* 444/*Array TYPE*



Declarations	445/Arrays as Parameters	445/Named Constants
in Pascal	447/Dynamic Arrays in Pascal	448
Warming Up to Pascal	448	
For Review	448/A Deeper Look	448/To Program 449
The Challenge in Pascal	450	
Case Study 5.4 in Pascal	450	
Case Study 5.5 in Pascal	451	
Case Study 5.6 in Pascal	452	
Case Study 5.7 in Pascal	454	
Case Study 5.8 in Pascal	456	
Working Out in Pascal	458	
<b>CHAPTER 6 CHARACTER DATA 459</b>		
6.1 GETTING ACQUAINTED	460	
Case Study 6.1 Input and Output of Strings	460	
The Solution — String Variables	460/Alternate Solution —	
Character Variables and Concatenation with the Header	Method 463/Alternate Solution — The Moredata Method	467/
Alternate Solution — The Trailer Method	467/	
Correctness	467/Efficiency 468/Summary — Case	
Study 6.1	468	
Case Study 6.2 Counting Vowels	469	
The Solution — Character Variables and the Case Statement	469/	
Correctness	471/Efficiency 471/Summary — Case	
Study 6.2	471	
Case Study 6.3 Reading and Printing a List of Names	471	
The Solution — String Arrays	472/Alternate Solution — Character	
Variables and Concatenation	474/Correctness 475/	
Efficiency	475/Summary — Case Study 6.3	475
6.2 IN RETROSPECT	476	
Character Variables and Constants	476/String Variables and	
Constants	477/String Operations 478/Input and Output of String	
Data	480/String Comparisons 481/Efficiency of String Handling	
Programs	482/Correctness of String Handling Programs	482
6.3 WARMUP EXERCISES	482	
For Review	482/A Deeper Look 483/To Program	483
6.4 THE CHALLENGE	484	
Case Study 6.4 Sorting Names	484	
The Solution — Insertion Sort	484/Correctness 485/	
Efficiency	485/Summary — Case Study 6.4	485
Case Study 6.5 Bank Compression	486	
The Solution — String and Character Manipulation	487/	
Correctness	492/Efficiency 492/Summary — Case	
Study 6.5	492	
Case Study 6.6 A Simple Word Processor	493	
The Solution — Text Manipulation	494/Correctness 498/	
Efficiency	498/Summary — Case Study 6.6	498
6.5 WORKING OUT	499	