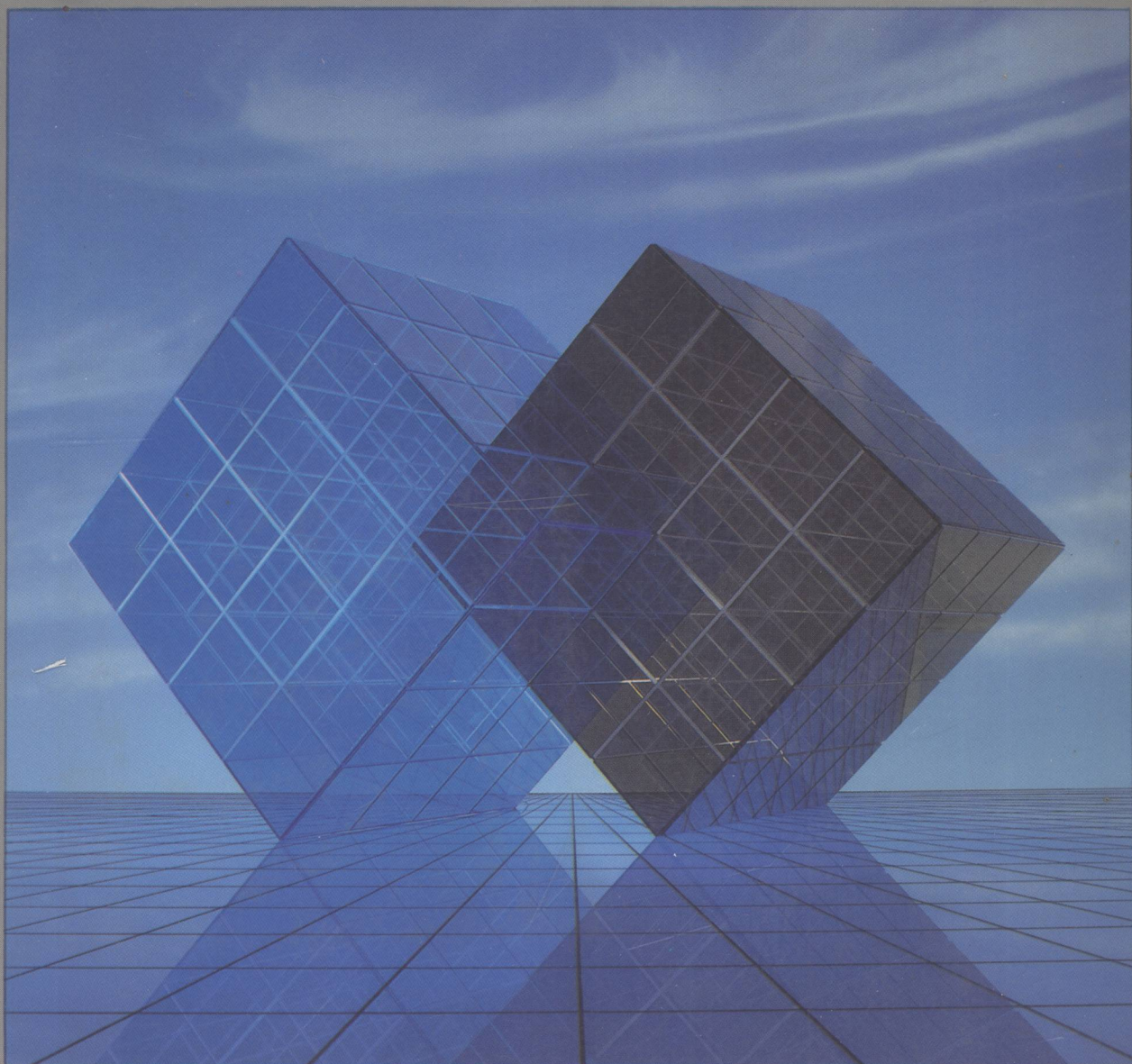

FUNDAMENTAL PROGRAMMING

WITH **FORTRAN 77** | A Science and
Engineering Approach



J. DENBIGH STARKEY
ROCKFORD J. ROSS

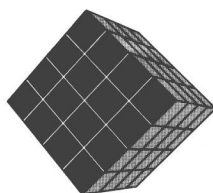
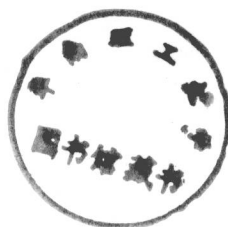
TP312
S795

9062538



Fundamental Programming With FORTRAN 77

A Science and Engineering Approach



J. Denbigh Starkey
Rockford J. Ross

Montana State University



E9062538

West Publishing Company

St. Paul • New York • Los Angeles • San Francisco

To our parents
Richard and Mary Starkey
Jim and Dorothy Ross

Production Management: Bookman Productions
Cover Photography: Michel Tcherevkoff for Alleghany
Cover Design: Hal Lockwood, Bookman Productions
Copyeditor: Janet Hunter
Compositor: G&S Typesetters, Inc.

Several trademarks and/or service marks appear in this book. The following companies are the owners of the trademarks and/or service marks following their names—
Apple Computer, Inc.: Apple; Commodore International Ltd.: Commodore; Digital Equipment Corporation: DEC; Hewlett-Packard Company: Hewlett-Packard; International Business Machines Corporation: IBM; Tandy Corporation: TRS-80.

COPYRIGHT © 1987 By WEST PUBLISHING COMPANY
50 W. Kellogg Boulevard
P.O. Box 64526
St. Paul, MN 55164-1003

All rights reserved

Printed in the United States of America

Library of Congress Cataloging in Publication Data

Starkey, J. Denbigh.
Fundamental programming with Fortran 77.

Includes index.

1. FORTRAN (Computer program language) I. Ross,
Rockford. II. Title.
QA76.73.F25S77 1987 005.26 86-18899
ISBN 0-314-77805-5

Preface

Fundamental Programming with Fortran 77 is the third book to appear in a series that includes *Fundamental Programming* and *Fundamental Programming with Pascal*. Our purpose for this book is to introduce Fortran 77 within the setting of a true course in programming. Besides giving students rudimentary programming skills and a knowledge of Fortran 77, such a course should leave students with a basic understanding of the science of programming.

The subtitle of our book, *An Engineering and Science Approach*, captures our belief that programming should be taught like other engineering and science courses with a primary emphasis on design and general scientific principles. Just as we would not hire an engineer to build a bridge who knew only the tools and structures used in bridge construction but nothing about bridge design, so we would not trust a programmer who knew only the statements and syntax of Fortran 77 but nothing about program design. Not only is an understanding of design issues crucial to safe and reliable products in both instances, but a knowledge of design methodologies provides a thorough understanding of the discipline in question. Most Fortran books do not provide this experience.

This book embodies an approach that is consistent with our responsibilities and objectives as computer scientists and computer engineers. Here, we first present the science of computing and the design issues fundamental to programming in sufficient depth. Then we describe how to implement well-designed and analyzed programs in Fortran 77. Students may not learn as many of the individual details of Fortran 77 in one term this way, but they will be markedly better programmers, and they will have the proper foundation for further individual study of programming.

We believe strongly in this method of instruction. Consider giving auto-mechanics' students a short course in the use of mechanics' tools, showing them how the tools work and having them practice tightening and loosening a few bolts. Then, a year later, give these same students a defective engine and say, "Fix it." Impossible? Of course. However, we would be guilty of a similar fault if we only presented the tools (Fortran statements) in this book without at the same time demonstrating the logical processes necessary for turning problems into well-designed programs. Thus, we emphasize program design, analysis, and verification as the most important aspects of programming. There are two primary benefits to this approach for the engineering, math, science, and business students who need a course in Fortran.

1. Engineering Design. A key issue in all engineering curricula is design. ABET, the Accreditation Board for Engineering and Technology, requires a substantial design component in all engineering disciplines. The emphasis of this book is approximately three-quarters program design (software engineering) and one-quarter Fortran implementation.

2. Advanced Study. A course taught with this book as its basis gives students a background that makes advanced study easier. While students may need to learn a different programming language (e.g., Pascal) as a prerequisite for advanced computer science courses, they will *not* need to relearn the issues fundamental to programming, making this transition to upper-level courses much easier. They can, in fact, usually learn the new language with little difficulty.

In determining the content of this book we had a definite pedagogical model in mind. This model stemmed from our realization a number of years ago that when we asked students to design a substantial program independently, many would return with a program consisting primarily of one or two large routines with little modularity; their programs were difficult to read and difficult to modify. Furthermore, when we asked the students simple questions about the efficiency or correctness of their programs, they 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, subroutines and functions were normally introduced late in these textbooks, almost as afterthoughts, as the "right way" to program. Students were mistakenly led to believe that subroutines 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:

PHILOSOPHY

1. Case Studies. The central pedagogical tools we use are case studies. 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. The case studies do not require mathematics beyond precalculus. Also, the case studies have been chosen to illustrate general program-design techniques.

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 details of Fortran 77, 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 Fortran 77). Each of the programs we design in the pseudolanguage is translated into a complete, working Fortran 77 program in a later section, where the new details of the Fortran 77 language are discussed separately from the problems involved in the program design.

3. Immediate Introduction of Subroutines and Functions. From the first case study on we teach that programs are collections of short, well-defined subroutines and functions, which are organized and called from an initial procedure (main program). The crucial concepts of subroutines, functions, parameters, and modular program design are thus ingrained into the habits and practice of students from the beginning. Students learn these topics without any 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 walk-through, semiformal verification steps (particularly for loops), program testing, selection of proper test data, robustness, and debugging techniques. Students receive a practical knowledge of the concepts of program verification.

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 case study as appropriate. 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; those not pursuing the subject after this course will understand practical methods for determining program efficiency.

ORGANIZATION

All chapters except the first follow a specific format designed to implement our philosophy. Each begins with three major sections: Getting Acquainted, In Retrospect, and The Challenge. 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 subroutines and functions developed there are often used in later case studies. In Retrospect summarizes these new concepts and provides a place to which to turn for review. The Challenge presents more challenging case studies involving the new concepts of the chapter. The Fortran implementation portions of each chapter mirror the previous sections exactly. In the Getting Acquainted with Fortran, Fortran in Retrospect, and The Challenge in Fortran sections, we translate into Fortran 77 and review the pseudolanguage programs of the case studies. All Fortran programs have been written to conform to the Fortran 77 standard.

Appendix A, Other Fortran Features, provides a concise reference manual for advanced Fortran 77 topics not covered in the text. 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 Fortran programs were tested on a VAX 11/750 computer using the Fortran 77 compiler developed by S. I. Feldman and P. J. Weinberger.

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 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 Fortran program is given in the exercises at the end of the chapter. The students can type this program in and run it 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 found its working version. We would be delighted to receive any comments you have, and corrections will be gratefully accepted and included in future printings or editions.

Special thanks go to Ruth Barton, Michigan State University, and Donna McClelland, Montana State University, who contributed their expertise to this book's development. Thanks are also due to the teaching assistants who used various forms of the Fortran portion of this book in teaching introductory Fortran courses at Montana State University: Mike Turner, Brian Thome, Bob Wall, and Jim Hill.

We would also like to thank those people who earlier reviewed the Fundamental Programming material that appears in this book: Gabriel Barta (University of New Hampshire); Rodney M. Bates (Kansas State University); Leland L. Beck (San Diego State University); Don Cartlidge (New Mexico State University); Cecelia R. Daly (University of Nebraska); Nancy Duffrin (SUNY at Stony Brook); Arthur C. Fleck (University of Iowa); Tamar E. Granor (University of Pennsylvania); James L. Hein (Portland State University); Rachelle Heller (University of Maryland); Leon Levine (University of California, Los Angeles); Gene Mahalko (University of North Dakota); Lawrence H. Miller (University of California, Los Angeles); Ralph Moore (Modesto Junior College, California); Keith R. Pierce (University of Minnesota); Alan L. Schwartz (University of Missouri, St. Louis); Robert F. Simmons (University of Texas, Austin); and Stephen F. Weiss (University of North Carolina, Chapel Hill).

Finally, Cheryl Ross, in addition to her responsibilities as wife and mother, cheerfully carried out the job of typing the manuscript.

IN GRATITUDE

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 have a sound understanding of the programming process. For example, it may well be that the most useful thing that a future engineer or scientist gains from this book is not a knowledge of Fortran 77 but an understanding of program correctness and efficiency, since the successful design of a program by members of a team or the speed of a particular software component of a system may be crucial in later projects. 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.

For highlighting concepts in the book we sometimes print words in *italics* or **boldface** type. *Italics* are reserved for phrases we want to emphasize and for terms that are being defined. **Bold** print is used in our program designs to mark keywords that are important. These uses of italic and boldface type will become clear as you read the book.

Whether you are a computer science student or a student from another discipline, this book will be useful to you now and later as a reference. One warning: if you have learned to program on your own, be careful! We have seen many sad cases of students with previous programming experience who started well but ended up doing poorly because they never shook off their earlier bad habits. If you use this book you will learn to be a competent programmer. We hope you enjoy learning to program.

J. Denbigh Starkey
Rockford J. Ross

Contents

Preface	xiv
To the Student	xviii

1	Getting Ready	1
1.1	BACKGROUND	2
1.2	THE MODEL COMPUTER	3
	The Processor	4
	The Store	8
	The Processor—Comparison Unit	16
	The Input and Output Devices	26
	Summary of the Model Computer	31
1.3	THE PROGRAMMING PROCESS	31
	Programs	32
	Functions and Procedures	32
	The Programming Process	36
1.4	EXERCISES	37
	For Review	37
1.5	FORTRAN IMPLEMENTATION	39
	Declaring Variables in Fortran	40
	Assignment Statements in Fortran	42
	Fortran Arithmetic Expressions and Operations	43
	Inputting Values in Fortran—The READ Statement	46
	Outputting Values in Fortran—The WRITE Statement	48
	An Example Fortran Procedure	48
1.6	FORTRAN EXERCISES	49

2	Simple Programs	53
2.1	GETTING ACQUAINTED	54
	Case Study 2.1: Compute the Average of Three Numbers	54
	Case Study 2.2: Area of a Rectangle	65
	Case Study 2.3: Swap Two Values	68

2.2	IN RETROSPECT	75
	Top-Down Program Design	75
	Functions and Procedures	75
	Program Structure	78
	Parameters	78
	Local Variables	86
	Program Verification	86
	Batch versus Interactive Programming	87
2.3	WARMUP EXERCISES	88
	For Review	88
	A Deeper Look	89
	To Program	91
2.4	THE CHALLENGE	91
	Case Study 2.4: Computing Simple Interest	92
	Case Study 2.5: Computing Compound Interest	94
	Case Study 2.6: Computing the Area of a Trapezoid	95
2.5	WORKING OUT	98
2.6	GETTING ACQUAINTED WITH FORTRAN	100
	Case Study 2.1 in Fortran	101
	Case Study 2.2 in Fortran	104
	Case Study 2.3 in Fortran	105
2.7	FORTRAN IN RETROSPECT	107
	Fortran Program Structure	107
	Fortran Initial Procedures	107
	Fortran Functions	109
	Fortran Procedures	111
	Fortran Variables—Some Problems	113
	Fortran Static Variables	115
	Choosing Fortran Variable Names	116
	Declaring Function Names in Fortran	116
	Fortran Parameters	116
	Integer and Real Numbers in Fortran	119
	Testing Fortran Programs	121
2.8	WARMING UP TO FORTRAN	121
	For Review	121
	A Deeper Look	122
	To Program	122
2.9	THE CHALLENGE IN FORTRAN	123
	Case Study 2.4 in Fortran	123
	Case Study 2.5 in Fortran	124
	Case Study 2.6 in Fortran	125
2.10	WORKING OUT IN FORTRAN	126

3 Making Decisions 127

3.1	GETTING ACQUAINTED	128
	Case Study 3.1: Find the Larger of Two Values	128
	Case Study 3.2: Compute the Largest of Three Values	131

Case Study 3.3: Sort Three Integers	140
Case Study 3.4: Assigning Student Grades	143
Case Study 3.5: Compute Ticket Costs	147
3.2 IN RETROSPECT	152
The If Statement	152
The Case Statement	155
Program Correctness	157
3.3 WARMUP EXERCISES	159
For Review	159
A Deeper Look	161
To Program	162
3.4 THE CHALLENGE	163
Case Study 3.6: Factorial	163
Case Study 3.7: Fibonacci Numbers	176
Case Study 3.8: Quadratic Equations	179
3.5 WORKING OUT	186
3.6 GETTING ACQUAINTED WITH FORTRAN	188
Case Study 3.1 in Fortran	188
Case Study 3.2 in Fortran	189
Case Study 3.3 in Fortran	194
Case Study 3.4 in Fortran	196
Case Study 3.5 in Fortran	198
3.7 FORTRAN IN RETROSPECT	201
The IF-THEN and IF-THEN-ELSE Statements in Fortran	201
The IF Statement with ELSEIF Clauses in Fortran	203
The Case Statement in Fortran	204
Creating Pleasing Output—the Fortran FORMAT Statement	206
Assertions in Fortran	227
Portability	227
3.8 WARMING UP TO FORTRAN	228
For Review	228
A Deeper Look	230
To Program	230
3.9 THE CHALLENGE IN FORTRAN	231
Case Study 3.6 in Fortran	231
Case Study 3.7 in Fortran	232
Case Study 3.8 in Fortran	232
3.10 WORKING OUT IN FORTRAN	233

4 Iteration 235

4.1 GETTING ACQUAINTED	236
Case Study 4.1: Sum a Series of Integers	236
Case Study 4.2: Sum the First n Even Integers	249
Case Study 4.3: Computing Average Scores	257
Case Study 4.4: Accumulating Interest	262
4.2 IN RETROSPECT	269
Loops	269

	Common Loop Operations	275
	Program Efficiency	276
	Determining Loop Correctness	293
4.3	WARMUP EXERCISES	307
	For Review	307
	A Deeper Look	308
	To Program	312
4.4	THE CHALLENGE	313
	Case Study 4.5: Factorial Revisited	313
	Case Study 4.6: Fibonacci Revisited	316
4.5	WORKING OUT	319
4.6	GETTING ACQUAINTED WITH FORTRAN	321
	Case Study 4.1 in Fortran	321
	Case Study 4.2 in Fortran	326
	Case Study 4.3 in Fortran	329
	Case Study 4.4 in Fortran	331
4.7	FORTRAN IN RETROSPECT	333
	Implementation of the General While Loop in Fortran	333
	Implementation of the While Moredata Loop in Fortran	337
	The Fortran do Loop	339
	Implementation of the Until Loop in Fortran	341
	The Fortran CONTINUE Statement	345
	The Fortran GO TO Statement	346
	Implementation of Assert Statements in Fortran	347
	Summary of Fortran Loops	347
4.8	WARMING UP TO FORTRAN	348
	For Review	348
	A Deeper Look	349
	To Program	350
4.9	THE CHALLENGE IN FORTRAN	350
	Case Study 4.5 in Fortran	350
	Case Study 4.6 in Fortran	351
4.10	WORKING OUT IN FORTRAN	352

5	Maintaining Simple Lists: Arrays of One Dimension	355
5.1	GETTING ACQUAINTED	356
	Case Study 5.1: Printing a List Forwards and Backwards	356
	Case Study 5.2: Computing Temperature Statistics	371
	Case Study 5.3: Computing Test Statistics	374
5.2	IN RETROSPECT	383
	Simple Lists	383
	The 1-D Array	384
5.3	WARMUP EXERCISES	389
	For Review	389
	A Deeper Look	390
	To Program	390

5.4	THE CHALLENGE	391
	Case Study 5.4: Computing the Dot Product of Two Vectors	391
	Case Study 5.5: Searching an Unordered List	394
	Case Study 5.6: Searching an Ordered List	400
	Case Study 5.7: Sorting a List of Values	409
	Case Study 5.8: Sorting a List of Values Faster	417
5.5	WORKING OUT	436
5.6	GETTING ACQUAINTED WITH FORTRAN	438
	Case Study 5.1 in Fortran	438
	Case Study 5.2 in Fortran	443
	Case Study 5.3 in Fortran	448
5.7	FORTRAN IN RETROSPECT	452
	1-D Arrays in Fortran	452
	Passing 1-D Arrays as Parameters in Fortran	453
	Fortran Parameters—a Second Look	454
	Implied Loops with Fortran READ and WRITE Statements	455
	Named Constants in Fortran—The PARAMETER Statement	457
	Dynamic Arrays in Fortran	458
5.8	WARMING UP TO FORTRAN	458
	For Review	458
	A Deeper Look	459
	To Program	459
5.9	THE CHALLENGE IN FORTRAN	460
	Case Study 5.4 in Fortran	460
	Case Study 5.5 in Fortran	461
	Case Study 5.6 in Fortran	463
	Case Study 5.7 in Fortran	465
	Case Study 5.8 in Fortran	467
5.10	WORKING OUT IN FORTRAN	467

6 Character Data 469

6.1	GETTING ACQUAINTED	470
	Case Study 6.1: Input and Output of Strings	471
	Case Study 6.2: Counting Vowels	478
	Case Study 6.3: Reading and Printing a List of Names	481
6.2	IN RETROSPECT	485
	Character Variables and Constants	486
	String Variables and Constants	487
	String Operations	488
	Input and Output of String Data	490
	String Comparisons	491
	Efficiency of String Handling Programs	491
	Correctness of String Handling Programs	492
6.3	WARMUP EXERCISES	492
	For Review	492
	A Deeper Look	492
	To Program	493

6.4	THE CHALLENGE	494
	Case Study 6.4: Sorting Names	494
	Case Study 6.5: Blank Compression	495
	Case Study 6.6: A Simple Word Processor	503
6.5	WORKING OUT	508
6.6	GETTING ACQUAINTED WITH FORTRAN	510
	Case Study 6.1 in Fortran	511
	Case Study 6.2 in Fortran	512
	Case Study 6.3 in Fortran	517
6.7	FORTRAN IN RETROSPECT	519
	Declaring Character Strings in Fortran	520
	Assigning Strings to Variables of Type CHARACTER	521
	Character String Parameters in Fortran	522
	Inputting Character Strings in Fortran	523
	Processing Text a Character at a Time in Fortran	524
	Fortran String Operations	525
	Variable Length Strings in Fortran	531
	The Fortran STOP and RETURN Statements	532
	The Fortran SAVE Statement	532
	The Fortran DATA Statement	533
6.8	WARMING UP TO FORTRAN	534
	For Review	534
	A Deeper Look	535
	To Program	535
6.9	THE CHALLENGE IN FORTRAN	536
	Case Study 6.4 in Fortran	536
	Case Study 6.5 in Fortran	539
	Case Study 6.6 in Fortran	543
6.10	WORKING OUT IN FORTRAN	547

7 Multi-Dimensioned Arrays 549

7.1	GETTING ACQUAINTED	550
	Case Study 7.1: Inputting and Outputting a Table	550
	Case Study 7.2: Operations on Tables	555
7.2	IN RETROSPECT	561
	2-D Arrays	561
	Arrays with More than Two Dimensions	562
7.3	WARMUP EXERCISES	563
	For Review	563
	A Deeper Look	564
	To Program	565
7.4	THE CHALLENGE	566
	Case Study 7.3: Multiplying a Matrix and a Vector	566
	Case Study 7.4: Multiplying Two Matrices	570
7.5	WORKING OUT	574

7.6	GETTING ACQUAINTED WITH FORTRAN	575
	Case Study 7.1 in Fortran	575
	Case Study 7.2 in Fortran	578
7.7	FORTRAN IN RETROSPECT	580
	Passing 2-D Arrays as Parameters in Fortran	581
	Input and Output of 2-D Arrays in Fortran	582
	Arrays of Higher Dimension	583
7.8	WARMING UP TO FORTRAN	583
	For Review	583
	A Deeper Look	584
	To Program	584
7.9	THE CHALLENGE IN FORTRAN	584
	Case Study 7.3 in Fortran	584
	Case Study 7.4 in Fortran	586
7.10	WORKING OUT IN FORTRAN	588

8 Records 589

8.1	GETTING ACQUAINTED	590
	Case Study 8.1: Constructing and Printing Employee Records	591
8.2	IN RETROSPECT	597
	Simple Records	597
	Arrays of Records	598
	Implementing Records Without the Record Data Type	598
	Correctness of Programs Using Records	601
	Efficiency of Programs Using Records	601
8.3	WARMUP EXERCISES	601
	For Review	601
	A Deeper Look	602
	To Program	602
8.4	THE CHALLENGE	603
	Case Study 8.2: Sorting Records	603
8.5	WORKING OUT	621
8.6	GETTING ACQUAINTED WITH FORTRAN	621
	Case Study 8.1 in Fortran	621
8.7	FORTRAN IN RETROSPECT	625
	Declaring Simple Records in Fortran	625
	Arrays of Records in Fortran	625
8.8	WARMING UP TO FORTRAN	626
	For Review	626
	A Deeper Look	626
	To Program	627
8.9	THE CHALLENGE IN FORTRAN	627
	Case Study 8.2 in Fortran	627
8.10	WORKING OUT IN FORTRAN	631

9	Constructing Arbitrary Data Structures— Linked Lists	633
9.1	GETTING ACQUAINTED	634
	Case Study 9.1: Input and Print a Linked List of Integers	643
	Case Study 9.2: Maintaining Several Order Lists	654
	Case Study 9.3: A Dynamic Waiting Queue	662
	Case Study 9.4: A Dynamic Priority Stack	673
9.2	IN RETROSPECT	679
	Constructing a Linked List	680
	Implementing Linked Lists with Arrays	681
	Passing Linked Lists Between Routines	681
	Queues, Stacks, and General Lists	682
	Correctness of Programs with Linked Lists	682
	Efficiency of Programs with Linked Lists	683
9.3	WARMUP EXERCISES	683
	For Review	683
	A Deeper Look	685
	To Program	685
9.4	THE CHALLENGE	685
	Case Study 9.5: A General Dynamic Order List	686
	Case Study 9.6: Nonrecursive Quicksort	698
9.5	WORKING OUT	709
9.6	GETTING ACQUAINTED WITH FORTRAN	710
	Case Study 9.1 in Fortran	711
	Case Study 9.2 in Fortran	721
	Case Study 9.3 in Fortran	725
	Case Study 9.4 in Fortran	733
9.7	FORTRAN IN RETROSPECT	738
	Linked Lists in Fortran	738
	The COMMON Statement	739
	The Fortran ENTRY Statement	744
9.8	WARMING UP TO FORTRAN	746
	For Review	746
	A Deeper Look	747
	To Program	747
9.9	THE CHALLENGE IN FORTRAN	748
	Case Study 9.5 in Fortran	748
	Case Study 9.6 in Fortran	756
9.10	WORKING OUT IN FORTRAN	762

10	External Files	763
10.1	GETTING ACQUAINTED	767
	Case Study 10.1: Construct an External File	767
	Case Study 10.2: Print an External File	772

Case Study 10.3: Searching an External File	775
Case Study 10.4: Deleting Records from an External File	783
10.2 IN RETROSPECT	788
Operations on External Files	788
Header Records, Trailer Records, and Checking for End-of-File	789
External Files as Parameters	790
Correctness of Programs with External Files	790
Efficiency of Programs with External Files	791
10.3 WARMUP EXERCISES	791
For Review	791
A Deeper Look	792
To Program	793
10.4 THE CHALLENGE	793
Case Study 10.5: Merging Two External Files	793
10.5 WORKING OUT	799
10.6 GETTING ACQUAINTED WITH FORTRAN	800
Case Study 10.1 in Fortran	801
Case Study 10.2 in Fortran	806
Case Study 10.3 in Fortran	809
Case Study 10.4 in Fortran	812
10.7 FORTRAN IN RETROSPECT	817
External Files on Real Computers	817
The Fortran OPEN Statement	819
Writing Sequential Files in Fortran	819
The Fortran ENDFILE Statement	822
Reading a Fortran Sequential File	822
The Fortran REWIND Statement	823
The Fortran CLOSE Statement	823
The Fortran BACKSPACE Statement	825
The Fortran INQUIRE Statement	825
Formatted Input in Fortran	826
Fortran DIRECT Files	828
A Reminder	829
10.8 WARMING UP TO FORTRAN	830
For Review	830
A Deeper Look	830
To Program	831
10.9 THE CHALLENGE IN FORTRAN	831
Case Study 10.5 in Fortran	831
10.10 WORKING OUT IN FORTRAN	835

Appendix A: Other Fortran Features	836
Appendix B: Answers to Selected Exercises	849
Appendix C: Fortran Intrinsic Functions	869
Index	872