
COMPUTING

A Problem-Solving Approach with FORTRAN 77

T. RAY NANNEY

COMPUTING

A Problem-Solving Approach with FORTRAN 77

T. RAY NANNEY

Professor of Computer Science
Furman University

Prentice-Hall, Inc.
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

Nanney, T. Ray

Computing: a problem-solving approach with
FORTRAN 77.

Includes index.

1. FORTRAN (Computer program language)
2. Electronic digital computers—Programming.

I. Title.

QA76.73.F25N36 001.64'24 80-28220

ISBN 0-13-165209-5

To

Elizabeth Nanney

who inspired this work and helped
with every phase of its development.

*Editorial/production supervision and
interior design: Service to Publishers*

Cover design: Lee Cohen

*Manufacturing buyer: Joyce Levatino and
Gordon Osbourne*

© 1981 by Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632

All rights reserved. No part of this book
may be reproduced in any form or by any means
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

PRENTICE-HALL INTERNATIONAL, INC., London
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, Sydney
PRENTICE-HALL OF CANADA, LTD., Toronto
PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi
PRENTICE-HALL OF JAPAN, INC., Tokyo
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., Singapore
WHITEHALL BOOKS LIMITED, Wellington, New Zealand

PREFACE

This book is a text for a beginning course in computer science in a liberal arts environment. The organization of topics in the book is the result of the continual evolution of a course that has been taught at Furman University since 1968. For the majority of students at Furman, this course would be the only one that they would take in computer science. Yet, many of them would later be required to write computer programs both in courses and in independent study in their own disciplines. It was essential, therefore, that the students have the opportunity to become good programmers. I had observed in previous introductory courses that many intelligent, motivated, nonscience students had great difficulty mastering the material in the traditional programming-oriented introductory course. I believed this problem to be a result of the teaching method used to present the material, not the inadequacies in the abilities of the students.

The format of the present course crystallized in the summer of 1975 during a six-week visit to the laboratory of Professor Seymour Papert of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. In his laboratory, I observed average 10- and 11-year-old children writing relatively complex programs in the computer language LOGO. This reinforced my belief that any motivated college student should be able to learn programming. Many of the techniques used in this book have their origin in ideas found in the work of Professor Papert.

The material in this book has been used in approximately 24 sections of the introductory computer science course during the period from 1975 to 1980. Although the evidence is rather qualitative, there appears to have been a significant improvement in the performance of students, especially nonscience majors, using the approach. Grades have improved, more programs have been completed, and students report having more fun than in previous years.

I am grateful to my colleagues James H. Keller and E. James Runde, who were willing to use this book while it was still evolving and who gave me valuable feedback. Three of my former students, C. Joseph Bridwell, James M. Coggins, and Cary A. Coutant, helped me in many ways. I am especially grateful to Seymour Papert, who gave me a stimulating place to work during the most crucial part of the writing.

T. RAY NANNEY

TO THE STUDENT

There are many approaches to the study of computing. This book emphasizes the writing of computer programs, but it also presents many other important aspects of computing. The course is essentially nonmathematical and is designed to be useful to both science and nonscience majors. No previous knowledge of computers, programming, or other specialized skills is required.

Regardless of your future interests and activities, it will be difficult (probably impossible) to avoid contact with computers. Consequently, in this book you will be introduced to programming, terminology, computer languages, applications, and social implications of computers. If you work conscientiously to master the material, at the end of the course you should be able to use the computer to solve problems that interest you. You should also have gained insight into many other important topics in computing.

The writing of computer programs is intimately associated with problem solving, so considerable attention will be given to a systematic approach to problem solving. At the end of the course many of you will find it easier to analyze and solve problems. A thorough understanding of the contents of the course will also improve your ability to think logically and critically.

An important side benefit of our study will be an enhanced understanding of the nature of language and the difficulties associated with using language in a precise, unambiguous manner. In fact, to solve a problem using a computer *requires* that language be used with precision. There is considerable truth to the statement that "computer science is more closely associated with linguistics and communication than with mathematics."

Studying computing can and should be exciting and fun. It is, however, quite different from anything you have studied previously; if you get behind, it can be almost impossible to catch up. It can also be a disastrous mistake to wait until the last minute to undertake a programming assignment. A problem that is quite easy for 95% of the class can be time consuming and difficult for you. The only safe assumption is that the problem will take longer than you had planned.

In writing computer programs, the following somewhat contradictory attitude is suggested: strive for perfection, but do not be upset or embarrassed by a mistake. When you write a program, you should try to write it so perfectly that correct answers are obtained when the program is first run on the computer. Yet it is not the nature of human beings to be perfectly accurate. So if you make a mistake, it is not a catastrophe—correct the mistake and rerun the program.

A major goal of this course is to learn to write computer programs using the FORTRAN language. Traditionally, in such a course you would learn a small part of the language and immediately begin programming. In contrast, we will not begin writing FORTRAN programs until an overview of computing has been presented and

x / TO THE STUDENT

some principles have been discussed. This approach appears to have the following advantages:

1. The introductory material will aid you in organizing your thinking about how to solve the problem for which you want to write a program. This is helpful for everyone and is especially valuable for students with nonscientific or nonmathematical backgrounds.
2. The initial study of programming principles helps you to develop and use good programming techniques from the beginning of the course. This is invaluable when the programs become longer and more difficult.
3. The knowledge of programming principles gained in the first part of the course makes a language like FORTRAN easier to learn and reduces frustration. The experimental evidence suggests strongly that students become better programmers because of the introductory material.

CONTENTS

Chapter

1	COMPUTERS AND COMPUTER SCIENCE	i
1.1	Computers in Society	1
1.2	The Speed of the Computer	3
1.3	The Decreasing Cost of Computers	4
1.4	The Computer as Problem Solver	4
1.5	High-Level View of Computer Structure	5
1.5.1	Some Components of the Central Processing Unit	7
1.5.2	The Input/Output Processor	9
1.6	Microcomputers and Minicomputers	10
1.7	The Reliability of Computers	11
1.8	Computer Science	13
2	COMMUNICATION, INFORMATION, AND LANGUAGE	17
2.1	A Model of Communication	17
2.2	Information	19
2.3	Features of Natural Languages	20
2.3.1	Complexity of Language	20
2.3.2	Redundancy	21
2.3.3	Ambiguity	22
2.3.4	Syntax, Semantics, and Logical Validity	24
2.4	Computer Languages	24
2.4.1	Machine and Assembly Language	25
2.4.2	High-Level Languages	26
2.4.3	The Need for Higher-Level Languages	27
2.4.4	Problem-Solving Domains—Various Higher-Level Languages	27
3	ALGORITHMS	32
3.1	Processes	32
3.2	Effective Procedures	35
3.3	Algorithms	36
3.4	Computability Theory	37

Chapter

4	SOME PROGRAMMING CONCEPTS	39
4.1	Concept of a Computer Program	39
4.2	Display Turtles	40
4.3	Definition Mode	43
4.4	Arguments	44
4.5	Names and Values	44
4.6	Bugs and Debugging	45
4.7	Subprocedures	47
4.8	Programming Examples	48
4.9	Planning and Top-Down Programming	51
5	COMPLEXITY AND COMPUTER PROGRAMMING	58
5.1	Human Limitations	58
5.2	Structured Programming	59
5.2.1	The Size of Complex Programs	59
5.2.2	Testing Complex Programs	60
5.2.3	Does Structured Programming Work?	61
5.2.4	Rules for Structured Programming	61
5.3	Stepwise Refinement of Programs	67
5.3.1	Program Development Language (PDL)	67
5.3.2	Examples of the Use of PDL	68
5.3.3	Flowchart Technique	74
6	OVERVIEW OF FORTRAN	80
6.1	The Historical Development of FORTRAN	80
6.2	Classification of FORTRAN Statements	82
6.3	Format of FORTRAN Statements	82
6.4	Mode of Variables	85
6.5	Rules for Naming Variables	87
6.6	Assignment Statements	89
6.7	Control Statements	90
6.8	Input/Output Statements	92
6.9	Subprograms	95
6.10	Sample Programs	96
6.11	Programming Errors and Debugging	111
6.11.1	Errors in Control Cards	112
6.11.2	Syntax Errors	113
6.11.3	Logic Errors	114

Chapter

7	MORE INPUT/OUTPUT	121
7.1	Simple FORMAT Statements and Reading of Data	121
7.1.1	Integer-Mode Specification	121
7.1.2	Real-Mode Specifications	124
7.1.3	Both Integer- and Real-Mode Specifications in a FORMAT Statement	126
7.1.4	Nonexecutability of FORMAT Statements	126
7.1.5	Summary of READ and FORMAT Statements	127
7.2	Simple FORMAT Statements and Output of Data	128
7.2.1	Carriage Control	129
7.2.2	Output of Values	135
7.3	Literal Data	137
7.3.1	The Apostrophe Technique	138
7.3.2	Skip Specification	140
7.3.3	Tabulation Specification	141
7.3.4	Hollerith Specification	142
7.4	Printing Reports	143
8	ARITHMETIC	161
8.1	Assignment Statements	161
8.2	Hierarchy of Operations	164
8.3	Mixed-Mode Statements	167
8.4	Intrinsic Functions	170
8.4.1	Characteristics of Intrinsic Functions	170
8.4.2	Examples Using Intrinsic Functions	173
8.5	Programmer-Defined Functions	176
8.5.1	Arithmetic Statement Functions	178
8.5.2	Functions	179
8.6	Arithmetic Errors	183
8.6.1	Integer Overflow	183
8.6.2	Overflow and Underflow for Real Numbers	185
8.6.3	Addition and Subtraction of Real Numbers	189
8.6.4	Inexact Representation of Real Numbers	190
8.7	Rounding	193
8.8	Other Errors	196
9	DECISION MAKING	206
9.1	IF . . . THEN . . . ELSE . . . ENDIF Statement (Block IF)	206
9.2	Logical Variables, Expressions, and Statements	219
9.3	ELSEIF Statement	231

Chapter

9.4	FORTTRAN Logical IF Statement	241
9.5	FORTTRAN Arithmetic IF Statement	244
9.6	Computed GO TO Statement	248
10	LOOPS	261
10.1	The Structure of Loops	261
10.2	DO . . . ENDDO Statement	263
10.2.1	Syntax of the DO . . . ENDDO Statement	263
10.2.2	Semantics of the DO . . . ENDDO Statement	266
10.2.3	A Sample Program	271
10.3	DO Statement	274
11	ONE-DIMENSIONAL ARRAYS	288
11.1	An Introductory Example	289
11.2	Characteristics of Arrays	292
11.3	The DIMENSION Statement	294
11.4	Input/Output of One-Dimensional Arrays	296
11.5	Programming Examples	304
11.5.1	Frequency Distribution	304
11.5.2	Prime Numbers, Sieve of Eratosthenes	308
11.5.3	Standard Deviation	313
12	SUBROUTINES	326
12.1	Defining Subroutines	326
12.2	Using Subroutines	331
12.3	Subroutine Libraries	335
12.4	Programming Examples	335
12.4.1	Sorting	335
12.4.2	Table Lookup: Payroll Calculation	351
12.4.3	Descriptive Statistics	359
13	TWO- AND THREE-DIMENSIONAL ARRAYS	372
13.1	Descriptions of Two- and Three-Dimensional Arrays	372
13.2	Input/Output of Two- and Three-Dimensional Arrays	377
13.3	Programming Examples	384
13.3.1	Student and Class Averages	384
13.3.2	Frequency Distributions	390
13.3.3	Questionnaires	397
13.3.4	Eight Queens Problem	407

Chapter

14	CHARACTER MANIPULATION	427
14.1	Declaration of Character Variables	427
14.2	Input/Output of Character Variables	428
14.3	Assignment Statements Involving Character Variables	430
14.4	Comparison of Character Variables	432
14.5	Programming Examples	435
14.5.1	Palindromes	435
14.5.2	Sorting Alphabetic Data	439
14.5.3	Translating Morse Code	441
14.5.4	Polish Notation	451
14.5.5	Manipulation of Strings	460
14.5.6	Finding Substrings—INDEX Function	471
15	SOME THINGS COMPUTERS CAN DO	483
15.1	Some Theoretical Results	484
15.1.1	Turing Machines	484
15.1.2	Self-reproduction of Computers	489
15.2	“Intelligent” Programs	490
15.2.1	Checkers	490
15.2.2	Chess	491
15.2.3	Geometric Analogy Problems	493
15.2.4	Learning a Concept	495
15.2.5	Natural Language	496
15.3	Thinking	500
15.3.1	The Imitation Game	500
15.3.2	An Experiment with an Imitation Game: PARRY	502
15.4	Education	504
15.4.1	The PLATO System	505
15.4.2	The LOGO Project	505
15.5	World Dynamics	506
16	SOCIAL ISSUES AND COMPUTING	511
16.1	Education	511
16.2	Limits to Growth	512
16.3	Privacy	513
16.4	The Cashless Society	516
16.5	Some Other Social Issues	518

Index

LIST OF FIGURES

Figure

1.1	Functional description of a computer	5
1.2	Some levels of knowledge for hardware and software	6
1.3	Major components of a computer	6
1.4	Functional description of a small time-sharing computer	10
1.5	A structure of computer science	13
2.1	General model of communication	18
3.1	Some flowchart symbols	33
3.2	Part of a flowchart for making French bread	34
3.3	Algorithm for determining whether or not an integer is a prime	37
4.1	CRT screen for display turtles	41
4.2	Different procedures for drawing a square	49
5.1	Illustration of the sequence structure	62
5.2	The IF structure	64
5.3	Use of the IF structure	65
5.4	The WHILE structure	65
5.5	Use of the WHILE structure	66
5.6	Program in flowchart form: class average for a quiz	75
5.7	Flowchart of a golf swing	76
6.1	Card layout for FORTRAN statements	83
6.2	Continuation cards	85
6.3	Payroll report	100
6.4	Manhattan Island problem	104
6.5	Prime numbers less than 1000—method 1	108
6.6	Prime numbers less than 1000—method 2	110
6.7	Steps in executing a program	112
7.1	Temperature conversion table	146
7.2	Payroll report	150
7.3	Manhattan Island problem	150
8.1	Computation of the hypotenuse of right triangles	175
8.2	Roots of the quadratic equation	176
8.3	Prime numbers less than 1000: illustration of MOD function	177

Figure

8.4	Roots of the quadratic equation: illustration of arithmetic statement functions	180
8.5	Simulation of rolling dice: illustration of a user-defined function	182
8.6	Calculation of pi	188
8.7	Comparison of summation techniques	190
8.8	Rounding and truncation	195
9.1	Calculation of the roots of the quadratic equation	213
9.2	Scoring of bowling	220
9.3	Scoring of bowling: illustration of logical variables	229
9.4	Quiz grades: illustration of ELSEIF statement	235
9.5	Quiz grades: using standard IF statements	236
9.6	Scoring of bowling: illustration of the ELSEIF statement	239
9.7	Quiz grades: illustration of FORTRAN logical IF statement	245
9.8	Scoring of bowling: illustration of arithmetic IF statement	249
10.1	Comparison of DO and WHILE structures	267
10.2	Three-digit numbers equal to the sum of the cubes of their digits	275
11.1	Frequency distribution of grades	306
11.2	Frequency distribution of grades: 10-point intervals	309
11.3	Prime numbers: sieve of Eratosthenes	312
11.4	Prime numbers: sieve of Eratosthenes—use of DO statement	314
11.5	Calculation of standard deviation: method 1	316
11.6	Calculation of standard deviation: method 2	317
12.1	Hypothetical description of a subroutine in a subroutine library	334
12.2	Sorting using additional memory: WHILE statement method	340
12.3	Sorting using additional memory: DO statement method	342
12.4	Exchange method for sorting: WHILE statement method	346
12.5	Exchange method for sorting: DO statement method	347
12.6	Improved exchange method of sorting	349
12.7	Payroll report program using binary search	357
12.8	Descriptive statistics	363
13.1	Student and class averages	389
13.2	Distribution of grades by student classification	393
13.3	Distribution of grades by student classification and sex	395
13.4	Sample questionnaire	397
13.5	Questionnaire summary	404
13.6	Eight queens problem	415
13.7	Eight queens problem: alternative solution	417

14.1	Palindromes	438
14.2	Alphabetic sorting: character variable method	441
14.3	Translation of Morse code	449
14.4	Conversion from infix to postfix Polish notation	457
14.5	Manipulation of strings using subprograms	472
14.6	Use of INDEX function	475
15.1	Diagram of a Turing machine	484
15.2	Typical problems solved by geometric analogy programs	494
15.3	Description of an arch	497
15.4	Scene in the blocks world	498
15.5	Abbreviated conversation in the blocks world	499
15.6	Arrangement for the imitation game	501
15.7	Partial interview with PARRY	503
15.8	Feedback loops of population, capital, agriculture, and pollution	509

LIST OF TABLES

Table

1.1	Number of computers used in the United States	2
1.2	Ownership of general-purpose conventional computers within the United States, 1976	2
1.3	Input/output devices	7
1.4	Comparison of the ENIAC and Fairchild F8 computers	11
2.1	Frequency of occurrence of characters of the English alphabet	22
2.2	Corresponding assembly- and machine-language instructions: $C = A + B$	26
2.3	High-level computer languages	28
4.1	Primitive instructions for turtle	41
5.1	Bugs observed in information bank programs for the <i>New York Times</i>	62
5.2	Relational operators	72
7.1	Carriage control indicators	130
8.1	Hierarchy of FORTRAN arithmetic operations	165
8.2	FORTRAN intrinsic functions	173
9.1	Definition of the logical operators	223
9.2	Hierarchy of logical operations	225
15.1	Summary of imitation game results for PARRY	504

CHAPTER 1 COMPUTERS AND COMPUTER SCIENCE

There is hardly an area of human endeavor that has not been affected by computers. Banks use computers to print your bank statements; utility companies compute your usage of their services and a computer prints your bill; computers are used as inventory control in department stores; airline personnel check your reservations using a computer. They are everywhere—in business, industry, government, education, sciences, and all present-day technology. This widespread use of computers in our society is due primarily to the generality of tasks that computers can perform, the tremendous speed of computer operations, and the decrease in the cost of computers.

1.1 COMPUTERS IN SOCIETY

In 1974, I heard an IBM executive refer to the rapid spread of computer usage as “the insidious revolution.” He explained that the general population rarely sees a computer, almost never works with a computer (computer billing being an exception), and is unaware of the diversity of computer applications and their pervasiveness in society. His claim was that without the public’s realization, we have become so dependent upon computers that society could not be maintained in its present form without them.

Let us expand our perspective regarding computer applications by briefly considering the history* of modern computing. The first large-scale electronic computer, ENIAC, was completed in 1946 and was used by the United States Army Ordnance Corps primarily to compute ballistic tables. In the early 1950s all computers were owned by the federal government and were used for census studies, weapons calculations, weapons delivery and control, cryptographic applications, nuclear design, nuclear engineering, inventory, and logistic applications. In 1954, the UNIVAC I computer was delivered to General Electric Company for commercial use, and the

*Much of this information has been abstracted from Ruth M. Davis, “Evolution of Computers and Computing,” *Science*, 195 (March 18, 1977), 1096–1102, copyright 1977 by the American Association for the Advancement of Science; and Saul Rosen, “Electronic Computers: A Historical Survey,” *Computing Surveys*, 1 (March 1969), 7–36. Copyright 1969, Association for Computing Machinery, Inc., reprinted by permission.

TABLE 1.1 *Number of computers used in the United States*

<i>Year</i>	<i>Number of computers</i>
1950	~12
1955	1,000
1960	6,000
1965	30,000
1976	220,000

TABLE 1.2 *Ownership of general-purpose conventional computers within the United States, 1976*

<i>Ownership by industrial classification</i>		<i>Percent of computers</i>
Manufacturing industry		31.0
Electric machinery	3.5%	
Nonelectric machinery	4.5%	
Other process manufacturing	9.7%	
Other manufacturing	11.0%	
Transportation equipment	2.3%	
Miscellaneous business		13.3
Advertising, employment, equipment, rental, engineering services, other professional services		
Banking, credit, insurance, real estate, and other financial institutions		13.4
Trade (wholesale and retail)		13.1
Educational institutions (schools, universities, libraries)		5.7
State and local government		5.7
Federal government		3.4
Transportation carriers		2.9
Medical and health services		2.7
Printing and publishing		2.4
Communications		1.9
Utilities (electric, gas, and sanitary services)		1.6
Other professional services		1.9
Petrochemical industry		1.0

Source: Ruth M. Davis, "Evolution of Computers and Computing," *Science*, 195 (March 18, 1977), 1100; by permission.

computer explosion had begun. The approximate number of computers in use in the United States at the end of various years is given in Table 1.1. In a 1967 report to the president of the United States,* it was estimated that 80,000 computers would be in use by the end of 1975. This estimate was far too low; improved computer tech-

**Computers in Higher Education*, Report of the President's Science Advisory Committee, The White House, Washington, D.C., February 1967. p. 58.