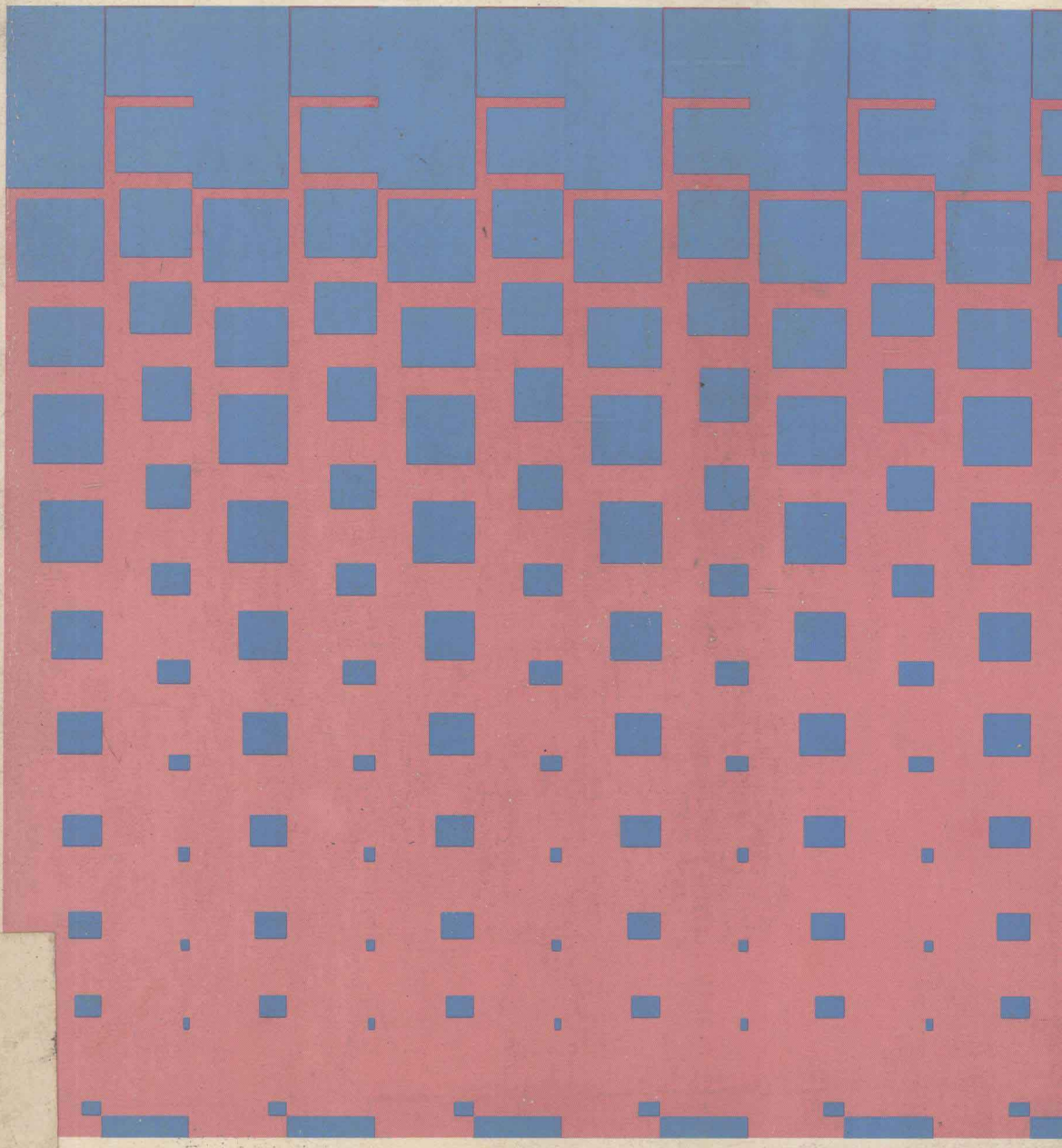
Rahul Chattergy
Udo W. Pooch

Top-down, Modular Programming in FORTRAN with WATFIV

Foreword by Gerald M. Weinberg



Winthrop Computer Systems Series

Top-down, Modular Programming in FORTRAN with WATFIV

RAHUL CHATTERGY

University of Hawaii

UDO W. POOCH

Texas A&M University

WINTHROP PUBLISHERS, INC.
Cambridge, Massachusetts

Library of Congress Cataloging in Publication Data

Chattergy, R.
Top-down, modular programming in FORTRAN with WATFIV.

Bibliography: p.
Includes index.
1. FORTRAN (Computer program language) 2. Modular programming. I. Pooch, U. W. II. Title.
QA76.73.F25C43 001.6'424 79-20802
ISBN 0-87626-879-3

© 1980 by Winthrop Publishers, Inc.
17 Dunster Street, Cambridge, Massachusetts 02138

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the publishers. Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Top-down, Modular Programming in FORTRAN with WATFIV

WINTHROP COMPUTER SYSTEMS SERIES

Gerald M. Weinberg, *editor*

SHNEIDERMAN
Software Psychology: Human Factors in Computer and Information Systems

GRAYBEAL AND POOCH
Simulation: Principles and Methods

WALKER
Problems for Computer Solutions Using FORTRAN

WALKER
Problems for Computer Solutions Using BASIC

CHATTERGY AND POOCH
Top-down, Modular Programming in FORTRAN with WATFIV

LINES AND BOEING
Minicomputer Systems

GREENFIELD
Architecture of Microcomputers

NAHIGIAN AND HODGES
Computer Games for Businesses, Schools, and Homes

MONRO
Basic BASIC

CONWAY AND GRIES
An Introduction to Programming:
A Structured Approach Using PL/I and PL/C, 3rd ed.

CRIPPS
An Introduction to Computer Hardware

COATS AND PARKIN
Computer Models in the Social Sciences

EASLEY
Primer for Small Systems Management

CONWAY
A Primer on Disciplined Programming Using PL/I, PL/CS, and PL/CT

FINKENAUR
COBOL for Students: A Programming Primer

WEINBERG, WRIGHT, KAUFFMAN, AND GOETZ
High Level Cobol Programming

CONWAY, GRIES, AND WORTMAN
Introduction to Structured Programming Using PL/I and SP/k

GILB
Software Metrics

GELLER AND FREEDMAN
Structured Programming in APL

CONWAY, GRIES, AND ZIMMERMAN
A Primer on PASCAL

CONWAY AND GRIES
Primer on Structured Programming Using PL/I, PL/C, and PL/CT

GILB AND WEINBERG
Humanized Input: Techniques for Reliable Keyed Input

试读结束，需要全本PDF请购买 www.ertongbook.com

To all of our ladies
who made our work necessary
yet worth the while

Foreword

It's become a cliché in computing that a new FORTRAN textbook must be justified in the light of the multitude of existing FORTRAN texts. Although I think the practice is sensible, *Top-down, Modular Programming in FORTRAN with WATFIV* needs no justification on those terms — neither from me nor the authors. Why? Because, as you'll notice, the book is "in FORTRAN," not "on FORTRAN." We don't have to justify a book being "in English," do we?

The book is in FORTRAN because a great many people "speak FORTRAN" and so wish to teach programming in that idiom. But the book is *about* programming, and that is what any such textbook should be about these days. We've long passed the time when we didn't know the difference between a language manual and a programming text.

At least we *should* have passed that time long ago, but many authors have not. Therefore, a book such as this — in which the authors pay scrupulous attention to style, design, and problem solving — is an extremely welcome addition to the textbook scene. The amount of care that Chattergy and Pooch have put into the choice and refinement of these examples is extraordinary. In their present state, the examples reflect many iterations of criticism and revision by students and colleagues — just what a hard-working textbook requires.

The choice of an introductory textbook always involves a great deal of personal taste on the part of the instructor or department. Students differ; schools differ; standards differ. But I'm confident that *Top-down, Modular Programming in FORTRAN with WATFIV* will strike a responsive chord in many places looking fo something beyond the run-of-the-mill FORTRAN language textbook — something that addresses both the instructor's needs and the students' needs in a unique style.

GERALD M. WEINBERG

Series Editor

Preface

The objective of this text is to show novice programmers general methods of program development, test, and modification using the FORTRAN language as a medium of expression. The language used is ANSI FORTRAN, with some features of the WATFIV version of FORTRAN incorporated into the earlier chapters to assist the novice programmer.

The spectrum of the FORTRAN language runs from ANSI FORTRAN to the various recently developed, structured versions of FORTRAN. To select the most useful version of FORTRAN, we considered some of the reasons for the popularity of FORTRAN in an age when much more powerful languages, such as PASCAL and PL/I, are readily available. We believe that one reason for this phenomenon is the existence of a large collection of FORTRAN programs in various areas of science and engineering. Another is the fact that FORTRAN is easy to learn and, consequently, has a large following among scientists, engineers, statisticians, and other numerical analysts. The versions of FORTRAN in use by these circles are very close to ANSI FORTRAN and nowhere near the structured versions of FORTRAN. FORTRAN '77 would have been our choice today if it had been invented in 1955. The time for structured FORTRAN has passed, since more powerful structured programming languages are already available. If structured programming were our goal, we would be wise to invest our time and efforts in learning PL/I or PASCAL and not tinker with yet another version of FORTRAN.

Then why write another text on FORTRAN? We find that FORTRAN is still the first programming language learned by many novice programmers. We also believe that bad programming habits picked up early have a very long half-life. We have seen many examples of this. One such case is that of a senior honors student in computer science who wrote a cross assembler for a microprocessor in FORTRAN where approximately every third statement was an arithmetic IF. Honors students in computer science presumably know everything about structured programming and other mathematical formalities of software engineering, but an arithmetic IF is still an old friend they can rely on in a pinch. Hence our purpose is to show that programs of reasonable

quality can be written, even in ANSI FORTRAN, by following some general principles of program development. In the past, the FORTRAN language has absorbed more than its share of blame for the lack of discipline and organization of the programmers using it. The recent rejuvenation of FORTRAN is yet another reason for this book. It appears that while we have been waiting for FORTRAN to fade away like an old soldier, it has started to appear on personal computer systems. Because of limited resources, most versions of FORTRAN on these systems are subsets of ANSI FORTRAN and lack the constructs of structured programming languages. If the potential for the use of the personal computer systems is fully realized, there will be thousands of programmers who will learn to program in FORTRAN for the first time. We hope that this text will provide them with a better introduction to programming in FORTRAN than many other books currently available.

The method for program development we have used is the top-down, modular method. This general approach to problem solving has been utilized for years; empires have been built on the principle of divide and conquer. One of the most concise descriptions of this approach is given by Hoare: "Inside every large problem there is a small problem struggling to get out" (q.v. *Playboy*, August 1978, p. 25). This method is illustrated by examples and augmented by the top-down testing and modification of programs. The top-down, modular method is in no way bound to any specific programming language. It is a way of organizing one's thoughts, always keeping one's goal in focus. The ultimate code may be more structured in PL/I than in ANSI FORTRAN, but the thought process that leads to this code must be organized and independent of the programming language. Whenever psssible, we have pointed out general concepts useful in programming, such as the importance of data structures, the perils of sharing data rather than procedures, and the usefulness of the principle of information hiding. Most texts on FORTRAN do not discuss these ideas, but they are important since they show many of the limitations of FORTRAN. We have, however, omitted the important topic of proving programs correct, since the mathematical background of most novice programmers does not justify its inclusion. In summary, our emphasis is on teaching the programmer the techniques of program development as early as possible.

We would also like to emphasize that this text is aimed primarily at amateur programmers. According to Weinberg,† the greatest difference between an amateur and a professional programmer lies in the ultimate clientele of the programs developed. A professional programmer never knows who the user(s) may be. He/she must survey every conceivable requirement of the users and protect programs from all modes of misuse. The amateur, on the other hand, programs for a small group of users and very often only for him/herself. Thus the programming environment of an amateur programmer is quite different from that of a professional. Fortunately for us, the majority of programmers are amateurs. We also believe that the difference between an amateur and a professional programmer is only one of degree and not of kind. Most professional programmers start out as good amateur programmers. Thus, although we have emphasized some simple techniques of defensive programming (checks for bounds violation of arrays, checks for simple errors in subprograms,

†Gerald M. Weinberg, *The Psychology of Computer Programming* (Princeton, N.J.: Van Nostrand-Reinhold, 1971), p. 122.

etc.), we have by no means discussed everything a professional programmer should do to ensure foolproof programs.

Most of our examples are selected to demonstrate basic programming techniques, such as the pairwise comparison or the binary search. The numerical examples are chosen in such a way that the underlying ideas can be simply explained by graphical means. We caution the reader that this is not a text on numerical analysis. We have discussed the mechanics of sorting, since the concept of sorting can be easily grasped without any profound knowledge of mathematics. The basic sorting techniques of exchange, insertion, and selection are demonstrated with simple examples. These are followed by more advanced methods of Shell's decreasing increment sort and the Quick sort. References are cited for the reader interested in their mathematical analysis. Discussion of random number generation includes a set of guidelines for designing such generators, and two specific models are given with references to their origin.

Many FORTRAN texts are written in sections starting with introductory concepts and progressing to advanced concepts. After some experimentation with this approach, we rejected it for its disadvantages. We found that programs written with only "elementary" concepts have awkward structures and, as such, are poor programs. Individuals becoming proficient in writing such programs have a difficult time breaking bad habits and making full use of the "advanced concepts" at a later date. For example, instead of

```
      IF(ERROR .LT. 0.0) ERROR = -ERROR
        IF(ERROR .LE. 0.0001) GO TO 100
- - - - - - - - - - - - - - - - - - - - - - -
100   STOP
```

we have used

```
      IF(ABS(ERROR) .LE. 0.0001) STOP
```

A person who does not understand the meaning of a simple function such as ABS will not find it easier to do so if it is simply postponed as an advanced concept relating to built-in functions. Similarly, a person who can visualize the flow of control during the execution of a program can understand the meaning of

```
      IF(DATA .GT. 0.0) SUM = SUM + DATA
```

when the flow of control is explained to him/her in the proper context. Treating such a statement as an advanced concept merely encourages the habit of branching at the least possible excuse.

We have omitted certain features of FORTRAN such as double precision, or complex and logical variables. Experience has shown that the beginner is rarely confronted with problems where these features are important and hence is not particularly motivated to learn them. We hope that when the need arises, the reader will be motivated to find these features in the FORTRAN manual published by his/her vendor.

If the reader has already visited the local computing center and fears for his/her sanity, these fears are fully justified. The computing center is a strange world of endemic chaos populated by demigods who speak the strange language of DD asterisks and DELETE, DELETEs. Unfortunately, the rituals vary from one center to the next and cannot be summarized in one text. It is on-the-job learning that the reader must acquire with whatever help is available. The reader will perhaps learn that computer science is somewhat like economics. The same computer print-out can be analyzed by different computer experts and result in different erroneous conclusions. The only note of encouragement that we can offer the reader is the motto of General Joseph W. Stilwell: "Illegitimati non Carborundum." Freely translated by the general, it means "Don't let the bastards grind you down."†

We will consider our efforts well spent if, at the end of this text, the reader comes to the following conclusions:

(i) The top-down, modular method is a wise approach to program development;
(ii) Programs of reasonable quality can be written even in ANSI FORTRAN; and
(iii) There is more to programming than can be discussed in a simple FORTRAN text.

If the reader does not share these views, let us disagree as friends and remember that this too shall pass.

RAHUL CHATTERGY
UDO W. POOCH

†Barbara Tuchman, *Stilwell and the American Experience in China* (New York: Macmillan Publishing Co., Inc., 1971), p. 5.

试读结束，需要全本PDF请购买 www.ertongbook.com

Top-down, Modular Programming in FORTRAN with WATFIV

Contents

1 Introduction to Digital Computers and Programming

Digital computers are widely used for solving problems in such areas as science, engineering, and business. This widespread use of computers is directly related to their ability to process rapidly and accurately enormous quantities of data following prespecified sequences of instructions. The process of creating these sequences of instructions is called *programming*, and the languages used for specifying these instructions are called *programming languages*. A complete sequence of data processing instructions considered as a single entity is called a *program*. In order to use a computer, we must have a thorough knowledge of at least one such programming language and the process of program development.

1.1 FUNCTIONAL DESCRIPTION OF A COMPUTER

The task of understanding what it means to program a computer and how it is programmed is simplified if we have some understanding of what a computer is and how it works. A detailed structural description of a computer is, however, beyond the scope of this text; fortunately it is not needed to program solution methods for computer applications. A functional description of a computer will be sufficient for our purpose (see Fig. 1.1). A computer contains a memory unit for storing information. A memory unit can be thought of as a linear arrangement of memory cells, each cell capable of storing information in coded form, using some internal computer code. The information stored in these cells consists of *instructions* and *data*. The processing unit consists of an arithmetic and logic unit and a control unit. It fetches instructions from the memory unit and executes them, in the process storing or fetching data in or out of the memory unit. The instructions normally specify, explicitly or implicitly, the locations of data in memory cells. The processor fetches instructions from *successive* memory cells, unless directed otherwise by special *branch instructions*. At this point, it is not necessary to know the details of the interactions among the pro-

试读结束，需要全本PDF请购买 www.ertongbook.com