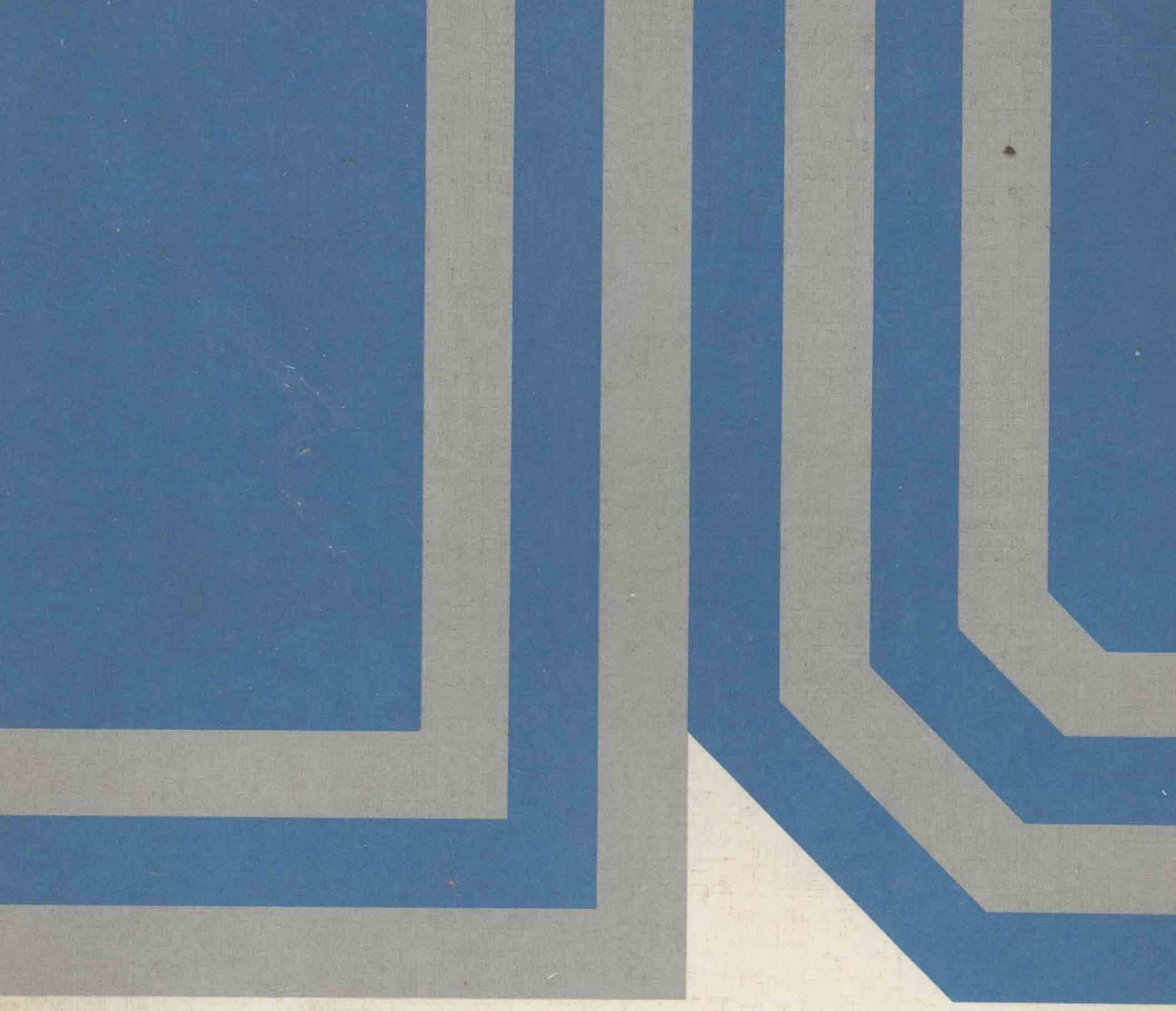

RICHARD E. HASKELL

ORTRAN PROGRAMMING

using structured flowcharts

FORTRAN PROGRAMMING USING STRUCTURED FLOWCHARTS

Richard E. Haskell

School of Engineering
Oakland University
Rochester, Michigan

SCIENCE RESEARCH ASSOCIATES, INC.
Chicago, Palo Alto, Toronto, Henley-on-Thames, Sydney, Paris, Stuttgart
A Subsidiary of IBM

Compositor and Illustrator Basil Wood

Acquisition Editor Robert Safran

Project Editor Jay Schauer

Cover Designer Judi McCarty

Printed in the United States of America.

Library of Congress Cataloging in Publication Data

Haskell, Richard E
FORTRAN programming using structured flowcharts.

Includes index.
1. FORTRAN (Computer program language) 2. Structured programming. I. Title.
QA76.73.F25H38 001.6'424 77-23931
ISBN 0-574-21135-7

10 9 8 7 6 5 4 3 2 1

To Jeff, Debbie, and Kim

PREFACE

FORTRAN has traditionally been taught in introductory computer courses. This practice has been due in part to the widespread availability of efficient FORTRAN compilers, the problem-solving capability of the FORTRAN language, and the position of FORTRAN as the earliest higher-level language in general use.

The introduction of the ideas of structured programming in recent years has led to criticism of the FORTRAN language as being unsuitable for structured programming. This notion is based on the fact that FORTRAN does not contain statements that are directly related to the more common control statements associated with structured programming. As a result of this situation a number of proposals have been made to add new "structured programming" statements to the FORTRAN language, and many structured FORTRAN preprocessors have been so implemented at various computing installations. There is, however, no uniformity to these preprocessors, and therefore one of the main advantages of FORTRAN *programs, namely their portability, is lost.*

The purpose of this book is to show that structured programs can be written in standard FORTRAN without the need for any preprocessor. These programs can be run with any of the present-day FORTRAN compilers, thus maintaining the primary advantage of portability. The key to writing structured

programs in FORTRAN is the use of structured flowcharts as an integral part of the program listing itself. Traditionally, the program flowchart and the program listing have been considered to be separate entities. The technique presented in this book for writing structured programs in FORTRAN begins by writing the program as FORTRAN arithmetic and logical expressions within the "boxes" that make up the structured flowchart. This form of the program can be written directly on a FORTRAN coding form beginning in column 21. The logic of the program is easily understood from this form of the program listing.

In order to make this form of the program executable using a standard FORTRAN compiler, the left-hand side (columns 1 through 21) must be filled in with appropriate FORTRAN control words. This mechanical process is easy to learn and takes little time once the program itself has been written on the coding form. The computer listing will contain only FORTRAN arithmetic and logical expressions in the spaces beyond column 21. The structured flowchart for the program can then be drawn directly on the program listing, thus obviating the need for a separate flowchart to complete the program documentation. This method has the advantage that the flowchart is never out-of-date with the current version of the program inasmuch as the program listing, in a sense, always carries its flowchart along with it.

Programs written using the techniques described in this book are always in a modular, well-structured form, thereby greatly increasing the readability of the programs and simplifying the process of program modification. I have arrived at the form of writing FORTRAN programs described in this book after several years of trying different methods for writing structured programs (including switching to ALGOL). I now prefer to write all of my own programs in FORTRAN using the method described in this book. I find the final version of the program that includes the structured flowchart to be more readable than even its ALGOL counterpart.

The sample programs presented in this book have all been class tested and have been used at one time or another as programming assignments in an introductory freshman computer course. In addition all of the material is covered in a 4-credit freshman course, *Introduction to Computer Science.* The course assumes no previous computer background but is intended for students who are majoring in computer science, engineering, mathematics, or a physical science. The students will write from 10 to 12 programs of their own throughout the course.

There has been considerable discussion about the best way to teach structured programming. Some teachers have advocated that students should write unstructured programs to begin with and then "convert" them to structured form. I completely disagree with this approach. Students do not have to be taught to think in an unstructured manner and to write unstructured and disorganized programs. Indeed, I have found that students develop good programming habits only if they have been exposed to well-structured programs from the beginning. The fact that this can be done using standard FORTRAN, as this book illustrates, makes the introduction of structured programming at the beginning of the first course in computer programming a realistic possibility.

The order of presentation of material in the book is designed so that a new and more advanced concept is introduced with each new sample program. The key to success in using this book is to study each sample program carefully and then to write similar programs based on the problems at the end of each chapter. I believe that people learn to program only by writing programs and that they learn to program *well* only by being continually exposed to well-written programs.

Chapter 1 introduces the basic ideas of structured programming and structured flowcharts. The three boxes that make up a structured flowchart (*sequence*, *iteration*, and *alternation*) are introduced, and the technique of developing a structured algorithm using a top-down programming approach is illustrated with a detailed example. The first FORTRAN programs are introduced in Chapter 2. These are simple input/output programs in which data is read from data cards and printed on the line printer. Some teachers may question this early introduction, but my experience has indicated that a couple of weeks of formatting input/output at the beginning of the course builds self-confidence and pays big dividends later on when the students are able to handle any input/output problem without considering it to be a major hurdle.

Chapter 3 introduces a FORTRAN construct that is often overlooked in introductory courses, namely, the end-of-file exit in a READ statement. Although this is not part of the ANSI-66 standard, it is available on most current FORTRAN compilers. This device is used to construct a READING iteration box that provides a convenient method for writing simple looping programs. Arithmetic operations and arithmetic expressions are also discussed in this chapter.

The intelligent use of logical variables is another topic that is crucial to writing well-structured programs but often is almost entirely neglected in beginning FORTRAN

courses. Chapter 4 begins with a discussion of logical expressions and logical variables and then develops the general DO WHILE iteration box. This is the "workhorse" looping box that is used in most of the sample programs. The DO UNTIL iteration box is also described in Chapter 4.

The introduction of the IF...THEN...ELSE alternation box in Chapter 5 completes the discussion of the basic structured flowchart boxes that can be used to write any computer program. Library functions and statement functions are also introduced in this chapter.

Subscripted variables and DO loops are introduced in Chapter 6. A sequential search algorithm and a straight insertion sort algorithm are included in the sample programs of this chapter. Subprogramming, including functions and subroutines, is covered in Chapter 7. The sample programs in this chapter include plot routines and an index sorting subroutine.

In Chapter 8 a large program for storing information concerning personal checking accounts is developed. This program illustrates the use of the top-down programming approach, multidimensional arrays, a CASE box, and the COMMON statement.

A method for using this book in conjunction with a FORTRAN preprocessor is described in Appendix A. FORTRAN statements that have limited use (or that should be avoided altogether) in writing structured programs are discussed in Appendix B.

Hundreds of students have taken an introductory FORTRAN course based on early and incomplete versions of this book. Their patience, cooperation, questions, and critical comments are gratefully acknowledged. In addition, I have benefited from many discussions about structured programming with David Boddy and Glenn Jackson of Oakland University, and Jim Elshoff of General Motors Research Labs. The suggestions of many reviewers, in particular Marilyn Bohl of IBM, have been very helpful. The loving support and understanding of my wife, Edie, was an essential ingredient to the completion of this book.

Richard E. Haskell
Rochester, Michigan

FORTRAN PROGRAMMING
USING
STRUCTURED FLOWCHARTS

TABLE OF CONTENTS

SAMPLE PROGRAMS

1 INTRODUCTION TO STRUCTURED PROGRAMMING

This book is concerned with solving problems on a digital computer. The reader will learn how to develop computer solutions to problems and how to code these computer programs in FORTRAN. The book's theme, *structured programming*, is a technique designed to make a good programmer better.

This chapter tells you the whats and whys of structured programming. It introduces the three basic structures from which all computer programs can be constructed. It then explains *top-down programming* by looking in detail at how a structured program can be written, using as an example the baking of peanut butter cookies. Finally, it defines the relationship between structured programming and the FORTRAN programming language. This latter topic will be our concern for the remainder of the book.

1.1 What is Structured Programming?

A digital computer is an electronic machine that can store information, transfer data from one location to another, and perform certain logical and arithmetic operations. The *hardware* of a digital computer consists of the electronic circuitry and related physical devices, whereas the computer programs that make this hardware perform useful tasks are called *software*.

In the early days of computers and continuing through most of the 1960s, the cost of the hardware was the dominant cost in computer systems. However, over the last several years the cost of software has exceeded the cost of hardware in most large computer organizations. This change in the cost balance between hardware and software is due to two main factors. On the one hand, hardware costs have decreased dramatically as the result of new technological developments. A major development has been the widespread use of large scale integrated (LSI) circuits. Using such techniques, very complex computer circuits can be mass produced at low cost. On the other hand, software costs are primarily personnel costs, and as everyone knows, these costs always increase.

Given this state of affairs, it is prudent to look for ways of decreasing software costs. One way is to increase programmer productivity. Studies have shown that a large fraction of the total programming effort is spent in debugging programs that don't work properly and in modifying existing programs. If programs were initially written in such a way that errors were minimized and the programs were easily modifiable by other people, this would be a large step in the direction of reducing software costs. These are some of the goals of structured programming.

Structured programming is basically a programming style that is designed to make the logic of programs more understandable, the code of programs more readable, and the execution of programs more reliable. The central feature of structured programming is the use of a minimum number of *basic building blocks* for which the logic is self-evident. Thus, the philosophy is to avoid tricky logical constructs and instead use simple logic that is easily understood by others, even if it results in slightly more computer time. Remember software (including software maintenance) now costs more than hardware. The three basic building blocks of structured programming will be described more fully in the next section.

Another characteristic of structured programs is their *modular structure*. If a change in one part of a program can inadvertently cause undesirable changes in other parts of the program, then program maintenance is a difficult problem. Thus, structured programs consist of many independent parts that are separated from each other to the maximum degree possible. The interconnections between these distinct parts must follow certain strict rules. By using this type of modular construction, the programmer can be sure that once one part of the program is coded and working properly, that part will not be affected by mistakes made in programming other

parts. This property of program modularity becomes very important in large programs.

The idea of *top-down programming* is also sometimes associated with structured programming. Top-down programming is much like outlining a report that you intend to write. First you write the title of the report, then the title of each chapter. For each chapter you list all subheadings, and then outline the contents of each of these subheadings. Finally you actually write some sentences for some part of the report. You do not necessarily have to start at the beginning, but the overall structure of the report should be clear in your mind before you begin to write any sentences. The technique of top-down programming is shown in Section 1.3 where we develop a structured program for baking peanut butter cookies.

In addition to structured coding using basic building blocks, program modularity, and top-down programming, structured programming has also come to mean anything that will reduce software costs. These topics have included a number of primarily managerial and organizational techniques, such as Chief Programmer Teams and Programming Production Libraries, which are beyond the scope of this book. Rather, we will focus our attention on how to formulate structured programs and code them in FORTRAN. In the process of doing this, the ideas of program modularity and top-down programming will develop naturally.

Since much of the impetus for structured programming comes from an attempt to reduce software costs, you may be saying, "But I don't plan to become a professional programmer; I just want to learn enough FORTRAN to write my own little programs that will only be run once and that no one else will ever read." (You would be surprised at all of the "own little programs that will only be run once" that are still being used today. You will be more likely to do small single-shot computations on small electronic calculators whose computing power has increased dramatically in recent years.) If you are going to take the time to write a FORTRAN program to be run on a general-purpose digital computer, you might as well take the time to write the program well and in such a way that if you later decide to modify it you will be able to do so easily.

But more important, a study of structured programming should give you a better appreciation of the inherent nature and structure of computer programs. It will help you to enhance your own skill and ability to solve problems. Let us therefore take a look at the three basic building blocks that are used in writing structured programs.

1.2 The Three Boxes of Structured Coding

It has been the custom in introductory computer courses to emphasize the use of *flowcharts* for solving problems. These flowcharts were designed to indicate the flow or logic of the computer program. They consisted of various shaped boxes interconnected by directed line segments. Since there were few restrictions on how the boxes could be interconnected, it was not unusual to wind up with a very complicated flowchart that was difficult to decipher. Such unstructured programs are not only difficult to understand, but are also difficult to modify and therefore costly to maintain.

As a result of this state of affairs, some advocates of structured programming have abandoned the use of flowcharts altogether, relying on the structured code itself to adequately describe the program. While it is true that the logic of a program should be apparent from the code of a well-structured program, experience has shown that beginning students of computer programming (and many advanced programmers as well) need some type of visual display that will communicate to them the logic of their computer programs.

The traditional type of flowcharts is replaced in this book with a new kind of structured flowchart that is made up entirely of the three basic "boxes" described below. There are a number of advantages to this approach. First of all, the computer solution for a particular problem can be obtained before it is necessary to worry about the details of coding the program in FORTRAN (or any other high-level language). Secondly, the use of structured flowcharts instead of the more traditional kind will force all programs to be structured programs and will encourage a top-down programming approach. Finally, it will be possible to superimpose the structured flowchart on the actual program listing from the computer. This means that a major part of the documentation of a program, including the flowchart that clearly indicates the logic of the program, will be included automatically in the computer listing of the program code.

A *structured flowchart* is made up entirely of the three basic building blocks of structured programming. These three building blocks are all in the form of boxes that can only be entered from the top and exited from the bottom as indicated in Figure 1.1. We will now describe these three boxes in more detail.