ADVANCED PROGRAMMING TECHNIQUES

A Second Course in Programming using FORTRAN

by

Charles E. Hughes
Charles P. Pfleeger
Lawrence L. Rose

published by

John Wiley and Sons, Inc.
New York London Sydney Toronto

** PREPRINT EDITION - 12/1/77 - PREPRINT EDITION **

** PREPRINT EDITION - 12/1/77 - PREPRINT EDITION **

ADVANCED PROGRAMMING TECHNIQUES

A Second Course in Programming using FORTRAN

by

Charles E. Hughes
Charles P. Pfleeger
Lawrence L. Rose

published by

John Wiley and Sons, Inc.
New York London Sydney Toronto

** PREPRINT EDITION - 12/1/77 - PREPRINT EDITION **

PREFACE to the preprint edition

This book is designed to cover material commonly presented in a second programming course. It includes the majority of topics currently being considered for inclusion in the second course of the Revised ACM Curriculum '68 Recommendations.

Our objective in writing this book was to bring together much of the material necessary for a novice programmer to mature into a professional. This aim is accomplished by discussing principles and then presenting examples which adhere to these principles. Most example programs presented here are non-trivial. In all cases, these programs are carefully documented, represent well-structured algorithms and have been extensively machine tested.

The programming language which we use to implement our algorithms is FORTRAN. We present specific details on the WATFIV dialect and on those dialects of 1966 ANS FORTRAN IV as implemented on the IBM 360-370 and DEC-10 series machines. Where appropriate, we discuss changes to FORTRAN IV which have been made in the new ANS proposed standard (FORTRAN 77).

We have written this book to be usable both by students of computer science and by major computer users. No extraordinary sophistication is assumed of our readers. The only background we require is programming experience equivalent to having successfully completed a basic course in some procedure-oriented language (not necessarily FORTRAN).

The material we present here is organized into nine chapters and two appendices. The first two chapters (0 and 1) form the introduction. In Chapter 0 we discuss that portion of the FORTRAN language which is assumed in subsequent chapters. The presentation here is brief, serving as a review for those already having a basic knowledge of FORTRAN and as a primer for the

reader who knows some other procedure-oriented language. In Chapter 1 we introduce topics of algorithm selection, program development, programming style, documentation standards, debugging techniques and program maintenance.

In Chapters 2, 3 and 4 we present advanced features of the FORTRAN language. In Chapter 2 we discuss subprograms, with a particular emphasis on argument passing techniques. In Chapter 3 we deal with character manipulation. We describe the organization of tapes and disks in Chapter 4 and present FORTRAN statements for utilizing these devices. In addition, we introduce some relevant features of the IBM OS/360 and DEC-10 TOPS command language.

Chapters 0 through 4 contain examples whose data structures are naturally represented by scalars and arrays. In Chapter 5 we present techniques for representing and manipulating complex data structures (linked lists, stacks, queues, trees).

In Chapters 6, 7 and 8 we describe aspects of the environment -- software and hardware -- in which a user program must reside. In Chapter 6 we discuss the representation of data within digital computers. We present elementary machine organization in Chapter 7. Finally, in Chapter 8 we discuss techniques for reducing the cost of program development, maintenance and execution. Specifically, we introduce our readers to the notions of object decks, program relocation, loading and overlay structures.

In Appendix 1 we summarize features of FORTRAN 77 which differ from those of FORTRAN IV as presented in the main parts of this text. Appendix 2 lists and defines each of the function subprograms which are standardly built into most FORTRAN dialects. [The appendices are not supplied with this preprint edition, but will be provided in the standard text.]

The authors wish to acknowledge the many helpful suggestions offered by Al Davis and our other three (anonymous) referees. We wish to thank the many students who braved their way through earlier versions of the text. Finally, we wish to express our gratitude to Joyce Marlar who skillfully typed much of the original manuscript.

All program presented here were tested using the IBM 360 model 65 at the University of Tennessee Computing Center. In addition, several of them were tested on the DEC-10 KL processor at the University of Tennessee and on the DEC-10 KA processor operated by the Ohio State University's Computer and Information Sciences Department.

This is a special preprint edition. The final version is presently being typeset, and should be available during the spring of 1978. The authors would appreciate any comments on this edition, especially those pointing out errors in the text. Any errors should be reported to

Charles E. Hughes
The University of Tennessee
Computer Science Department
Knoxville, Tennessee 37916
phone (615) 974-5067

We appreciate your using this version.

Charles E. Hughes
Charles P. Pfleeger
Lawrence L. Rose

12/1/77

CHAPTER 0

PRELIMINARIES

0.1 OVERVIEW

This book will help you to become a mature programmer. In it you will see examples of the tools and the techniques that experienced programmers use in solving their problems. We have chosen FORTRAN as the basis for our discussion, but much of what you will learn here also applies to other languages. We will describe many properties of computing, independent of the language being used.

As programmers mature, they develop "style," which means that they adopt certain habits of program structure, documentation, and coding. Chapter 1 is devoted to programming style, and we will also note points of good style throughout this text. In Chapters 2 to 4, you can learn about topics such as subprogram usage, character manipulation, and tape and disk I/O. In later chapters you will learn about the environment in which your program runs; we describe the internal representation of data, the structure of a digital computer, and the process of program execution.

Perhaps you have never learned FORTRAN, or your exposure to it occurred some time ago. This book has been designed for use by anyone with experience in some procedure-oriented language. If you know a language such as !PL/I, !ALGOL, !PASCAL or !COBOL, you should have no trouble learning FORTRAN. This preliminary chapter presents the material that you will need later. Even if you already know FORTRAN, we suggest that you read this chapter to refresh your memory and to guarantee that you know all of this background material. The description of each statement has been condensed to a block and shaded so that you can refer to it quickly.

In this text we describe FORTRAN as standardized by the American National Standards Institute in report X3.9-1966. A new standard has been proposed and appears to have achieved a favorable response; this new version is called FORTRAN 77. In places where the new form differs from the standard, we will mark that section with a vertical bar in the margin; you should refer to Appendix 2 for a description of the change.

Some features of FORTRAN depend on which computer and which compiler are being used. Some compilers accept part or all of standard FORTRAN, and some allow extensions to the standard. We will call each combination of a machine and a compiler a <u>dialect</u>. Many of our dialect-dependent examples are based on two widely used dialects: the IBM 360-370 computers under the WATFIV compiler, and the DEC 10 computer with the compiler FORTRAN-10.

0.2 BACKGROUND AND DEFINITIONS

0.2.1 Statement Form

A FORTRAN program consists of a series of statements; these statements appear in a fixed form on input records with at most one statement per record. Each statement looks like the punched card in Figure 0-1. Although the actual input medium could be something else, such as a line of a typewriter terminal or a record on a paper tape, we will speak as if the input comes from cards.

Figure 0-1 FORTRAN Statement Format

A statement number in FORTRAN is a number appearing anywhere in columns 1 to 5. A statement number is used on a statement that needs to be related to other statements in the program. On most FORTRAN statements, the number may be omitted.

The body of a statement is punched in columns 7 to 72. Blanks are ignored in the statement body (except within character data, to be described shortly); this implies that the body need not begin exactly at column 7, and that blanks may be inserted as desired to improve readability.

If the body of one statement is too long to fit on a single card, the statement may be continued by punching any character, except a blank or zero, in the continuation field (column 6) of the next card, and continuing the statement in columns 7 to 72 of this new card. Additional continuations may be made by punching a nonblank character in column 6 of succeeding cards. A maximum of 20 cards (19 continuations) is allowed for any one statement.

Columns 73 to 80 are reserved for a sequence number, although this may be omitted. (For some input forms other than cards, this field does not exist, or it may be filled in for you.) On large programs it is a good idea to punch numbers in this field to help you sort the input deck if it should be dropped.

Column 1 is used to identify comments. Any card with the letter C in column 1 is taken as a comment and is ignored by the compiler. Comments are used to identify a program, to separate a program into logical segments, or to explain a particular segment of a program.

TABLE OF CONTENTS

CHAPTER 0

PRELIMINARIES

0.1 OVERVIEW

This book will help you to become a mature programmer. In it you will see examples of the tools and the techniques that experienced programmers use in solving their problems. We have chosen FORTRAN as the basis for our discussion, but much of what you will learn here also applies to other languages. We will describe many properties of computing, independent of the language being used.

As programmers mature, they develop "style," which means that they adopt certain habits of program structure, documentation, and coding. Chapter 1 is devoted to programming style, and we will also note points of good style throughout this text. In Chapters 2 to 4, you can learn about topics such as subprogram usage, character manipulation, and tape and disk I/O. In later chapters you will learn about the environment in which your program runs; we describe the internal representation of data, the structure of a digital computer, and the process of program execution.

Perhaps you have never learned FORTRAN, or your exposure to it occurred some time ago. This book has been designed for use by anyone with experience in some procedure-oriented language. If you know a language such as !PL/I, !ALGOL, !PASCAL or !COBOL, you should have no trouble learning FORTRAN. This preliminary chapter presents the material that you will need later. Even if you already know FORTRAN, we suggest that you read this chapter to refresh your memory and to guarantee that you know all of this background material. The description of each statement has been condensed to a block and shaded so that you can refer to it quickly.

In this text we describe FORTRAN as standardized by the American National Standards Institute in report X3.9-1965. A new standard has been proposed and appears to have achieved a favorable response; this new version is called FORTRAN 77. In places where the new form differs from the standard, we will mark that section with a vertical bar in the margin; you should refer to Appendix 2 for a description of the change.

Some features of FORTRAN depend on which computer and which compiler are being used. Some compilers accept part or all of standard FORTRAN, and some allow extensions to the standard. We will call each combination of a machine and a compiler a dialect. Many of our dialect-dependent examples are based on two widely used dialects: the IBM 360-370 computers under the WATFIV compiler, and the DEC 10 computer with the compiler FORTRAN-10.