Apple® FORTRAN

(Detailed Language Instructions
Specifically for the Apple® Computer)

by Brian D. Blackwood and
George H. Blackwood

Apple® FORTRAN

(Detailed Language Instructions
Specifically for the Apple® Computer)

by Brian D. Blackwood and
George H. Blackwood

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
FIRST PRINTING — 1982

International Standard Book Number: 0-672-21911-5
Library of Congress Catalog Card Number: 81-86556

Edited by: *C. W. Moody*
Illustrated by: *R. E. Lund* and *Wm. D. Basham*

Printed in the United States of America.

Preface

This book is written for the beginning programmer using the FORTRAN language on the Apple computer. Specific details are given for the use of each keyword, column relationship of the FORTRAN language, program statements, edit descriptors, and block statements. The book gives a brief history of computers, the development of the FORTRAN language, an introduction to the Pascal operating system, and the use of FORTRAN on single- and multiple-disk drive systems.

Each new topic is presented in clear and concise details. There are numerous programs that use these details so all aspects of the language are presented. The details of the language are broken down to the simplest form, and from this form the details are incorporated into the programs. The programs range from the elementary level, to demonstrate how the specific details of the language are used, to an advanced level useful in the business and engineering world.

There are numerous illustrations that demonstrate the specific parts of the language. The text and illustrations are coordinated to make the learning process as easy as possible. Should there be a difference of interpretation between the text and an illustration, the information in the illustration is the best authority. The interpretation of text and semantics can vary when approached from different points of view. The interpretation of an illustration may give a better understanding than does the text.

Originally, FORTRAN was developed for use by the scientific and engineering professions. Business is now using FORTRAN more because the language has so many features that are useful in solving business problems. FORTRAN certainly has the necessary computational features for handling business problems, and it also has formatting flexibility to produce business reports. There are numerous tested and cataloged function and subroutine subprograms available.

This book deals primarily with the FORTRAN language and is not intended as a programming manual. The flowcharts of some programs are given to introduce the logic of programming, but the problem-solving aspect of programming is not emphasized.

The problem-solving aspect of programming involves five points — (1) the problem must be completely understood, (2) the problem can be solved by the programmer using a pencil and paper, (3) the different conditions and cases relating to the problem must be completely understood, (4) suf-

ficient checks should be built into the program to prevent input data that is out of the legal range, and (5) the solution by the program must produce the correct results.

There is a story about the student who was complaining to his computer science professor about a failing grade on a program. The student said the program was beautifully written, well documented, and the printout was correctly formatted — only the output was incorrect. The professor replied, "I can get the wrong answer without writing a program."

BRIAN D. BLACKWOOD
GEORGE H. BLACKWOOD

Contents

CHAPTER 1
Introduction to Apple FORTRAN

The first electronic computers were developed around 1945 and 1946. The language used by these computers was in the binary (base-2) number system, and programs were very difficult to code and debug. In the period of 1951 to 1958, the UNIVAC computer was introduced and was programmed in machine language, which was assembled into binary. The machine language was easier to program because it used the hexadecimal (base-16) number system. In the period of 1958 to 1964, the batch-processing computers were introduced and they were programmed in assembly language. Assembly language is coded in mnemonics that resemble English words. Each higher level of language made communication between the computer and the programmer clearer and more understandable.

In 1954, John W. Backus at IBM began development of a FORTRAN compiler. The compiler translates FORTRAN source code into machine object code. FORTRAN and the FORTRAN compiler were further developed until the data-communications computers began using FORTRAN as a higher-level language for engineering and scientific purposes. FORTRAN was used with keypunched cards as the primary source of input to the computer.

The acronym *FORTRAN* stands for *FOR*mula *TRAN*slator. FORTRAN is a high-level problem-oriented language that can easily represent mathematical formulas. FORTRAN has great problem-solving capabilities for engineering and science, and the output is related to the problem. The output is not as "pretty" as it is when using other languages.

When the information-handling computers and the executive-aid computers were introduced, the computer terminal was used for input and output. With the use of the computer terminal as the primary source of input and output, keypunched cards were no longer necessary to input programs and data. FORTRAN was adapted for use with the terminal.

Apple FORTRAN is a subset, or dialect, of ANSI (American National Standards Institute) Standard FORTRAN, and is called Apple FORTRAN 77.

Although keypunched cards are not used with a terminal, FORTRAN programs are coded according to the columns on Hollerith (or IBM) cards. Each segment of the program statement must begin and end in the proper card column. For this reason, the relationship between Hollerith cards and the computer terminal is very important and will be referenced frequently.

Since cards are not used for the program or the input of data into the program, the data input into terminal-generated programs is entirely different.

There are three ways to input data into a terminal-generated program — (1) the data is placed in data statements within the program, (2) the data can be input from the keyboard into a compiled and linked program and produce output while the program is being executed, and (3) the data can be input from files on magnetic disks to produce output while the program is running.

The greatest change from using FORTRAN with keypunched cards and FORTRAN on a terminal is with the READ and DATA statements. The other aspects of FORTRAN on cards and FORTRAN on the terminal of the Apple microcomputer are very similar.

Integer BASIC and Applesoft use an interpreter to convert program statements into machine code during the live execution of the program. The program statements are translated by subroutines contained in read-only memory (ROM) into machine-code instructions. The interpreted statements are immediately used to run the program and to process data and information.

FORTRAN uses a compiler to convert source (initial) coded statements into machine-language object (final) code. The compiled source statements are called object code, and the object code is read and acted on by the computer.

For a source program to run, the computer must first translate each source-code language statement into its machine-code equivalent. The computer must also incorporate into the object program any library subroutines requested by the programmer and required by the computer. In addition, the program must be supplied with interconnecting links between different parts of the program.

A compiled program runs faster than an interpreted program; however, errors are more difficult to change and correct on a compiled program.

Each FORTRAN compiler is written for a specific computer and must convert the source-code program into object code for the specific computer. FORTRAN source programs are generally independent of the computer on which they will run, as long as the syntax between the language and the compiler is compatible. Many of the FORTRAN programs in this book were originally written to be run on a Honeywell 6000, and only a few changes were needed to make the programs execute on the Apple computer.

Source statements are written by the programmer to be converted to machine code by the compiler and linked to the system library so the program can execute.

Each FORTRAN statement is classified as executable or nonexecutable. An executable statement generates machine-code instructions that are executed by the computer. Nonexecutable statements are those statements that specify instructions to the compiler, and do not generate machine code or comments.

SOURCE STATEMENTS

There are seven types of source statements used in FORTRAN.

Executable Statements

1. Input/Output Statements -- These statements are used to control input and output devices and to transfer data and information between memory and input and output devices. READ and WRITE statements are examples of executable input and output statements.

2. Assignment Statements — Assignment statements assign a value to a storage location. The equals sign (=) following a variable assigns the value on the right side of the equals sign to the variable on the left side. The equals sign is used to provide the formulas needed for solution of problems, but has a different computer evaluation than in the mathematical usage.

3. Logical and Control Statements — These statements are used to control the order of execution of the program statements, and to terminate the execution of the program. Such statements include ELSE, ELSEIF, IF, GOTO, STOP, END, DO, and CONTINUE.

4. Subprogram Statements — These statements are used to execute subprograms, FUNCTION and SUBROUTINE, and to return from the subprograms. Such statements include CALL and RETURN.

Nonexecutable Statements

5. FORMAT Statements — These statements are used to shape the input and output statements, and control the location, form, and type in which data appears in the input and output. The keyword FORMAT appears at the beginning of each statement.

6. Specification Statements — These statements provide the FORTRAN compiler with information that is needed when the program is compiled. Specification statements initialize storage location to specified values, describe the limits of certain storage locations, establish array address locations, and supply other data and information about the program. Keywords include COMMON, DATA, DIMENSION, EQUIVALENCE, INTEGER, LOGICAL, and REAL.

7. Subprogram Statements — These statements include the FUNCTION and SUBROUTINE statements.

Fig. 1-1 is a FORTRAN program to show some of the types of executable and nonexecutable source statements.

A FORTRAN program consists of a series of FORTRAN statements (Figs. 1-1 and 1-2). FORTRAN statements are comprised of different elements that must conform to a consistent pattern for the program to be compiled and

```
C234567
      PROGRAM SUM3                          Comment statement*   (1)
      DIMENSION SCORE(5)                    Program statement    (1)
      DATA SCORE/89.,75.,100.,69.,47./      Specification statement (1)
      SUM = 0.                              Data input into program
      DO 20 J = 1,5                         Assignment statement (2)
      SUM = SUM + SCORE(J)                  Control statement    (2)
20    CONTINUE                              Assignment statement (2)
      WRITE (0,50) SUM                      Control statement    (2)
50    FORMAT (1X,//,' SUM = ',F10.0//)      Output statement     (2)
      AVG = SUM/5.                          FORMAT statement     (1)
      WRITE (0,60) AVG                      Assignment statement (2)
60    FORMAT (1X,' AVERAGE = ',F10.2//)     Output statement     (2)
      STOP                                  FORMAT statement     (1)
      END                                   Control statement    (2)
                                            Specification statement (1)
```

(1) Nonexecutable statement
(2) Executable statement

Fig. 1-1. FORTRAN statements.

executed. Any element that does not conform to the language standard causes a compile error, or run-time error.

Fig. 1-2A shows how data is placed into a FORTRAN variable. The keyword DATA is the beginning of the statement and is followed by the array variable (SCORE). A slash separator indicates that the values between the beginning and ending slashes are related to the array (SCORE), and conform to the dimension statement, DIMENSION SCORE (5). The numbers are reals, as indicated by the decimal following each number. Each real is separated by a comma (carriage return) to indicate the end of the value, and the last slash indicates the end of the list.

Fig. 1-2B shows an assignment statement. The values of the array SCORE are to be summed and assigned to the variable SUM. To make sure the first value in the address location SUM is zero, the variable SUM is initialized to zero.

The statement in Fig. 1-2C is the head of a loop control statement. The values in the loop must be integers and must relate to an integer loop variable. The value 20 in the DO loop statement relates to statement number 20, CONTINUE, which is the foot of the loop control statement (Fig. 1-2D).

Fig. 1-2E is a statement that outputs the value held in the address location AVG. This WRITE statement is written to the CRT (0 is the CRT), and uses the statement numbered 60 to format the output.

Fig. 1-2F is a FORMAT statement that prints AVERAGE = on the screen, and the AVG value is placed in a floating point (F) field that has 10 places in the total field width, and 2 places (F10.2) after the decimal point.