Fortran for Students

Roger Hutty

Fortran for Students

Roger Hutty

School of Mathematics, Computing and Statistics,
Leicester Polytechnic

First published 1980 by
THE MACMILLAN PRESS LTD
London and Basingstoke
Associated companies in Delhi Dublin
Hong Kong Johannesburg Lagos Melbourne
New York Singapore and Tokyo

Printed in Hong Kong

ISBN 0 333 25331 0

Preface

This learning text, based on FORTRAN 66, - the ANSI X3.9-1966 FORTRAN standard enables a student to develop through suitable exercises and programming practice a professional approach to programming and produce a work of quality. There is an art to programming: it is not enough for a student to know the rules of a language, it is equally important to develop a readable and intelligible program which has style and uses the best techniques.

The text is in the form of short narratives; each narrative covers one or two concepts and is followed by an exercise. The narratives move fast, the exercises are challenging and they encourage students to make many discoveries for themselves, thereby allowing the rapid development of skills. Every opportunity is taken to expose practical problems and develop good programming habits right from the start.

The text includes some opinions and opposing views for discussion.

Most of the facilities specified in the FORTRAN 66 standard are included in the text.

The main changes specified by the FORTRAN 77 (ANSI X3.9-1978) standard are included in the text as amendments at the end of each chapter. This way of including FORTRAN 77 was chosen because it will be several years before all of the facilities specified in the standard are available on all computer systems.

The text can be used with the computer systems of all the major manufacturers.

It is suitable for BSc and HND courses in Computing, Mathematics and Engineering and as a reference book for practising FORTRAN programmers.

NOTES FOR TUTORS

The text is divided into chapters. A chapter can be used as a self-contained unit for teaching purposes to be used for one or two weeks' lectures and tutorials or one self-instructional unit.

Each chapter assumes a student has satisfactorily completed the previous chapters.

The book is not an ordinary text book - it is a learning text. The narratives, the exercises, the exercise answers and the program at the end of each chapter all play an equally important part in the learning process. The narrative is fast moving. The exercises have been chosen to be a real test of the student's understanding of both concept and detail and to extend, sometimes by discovery, the knowledge and understanding gained in the narrative. Some of the exercise answers include notes on the answers which also extend a student's knowledge and understanding. A program is specified at the end of each chapter for the student to code, run and test on a computer system. Each program is designed, as far as is possible, to include the concepts contained in the chapter.

Students should be encouraged to do the exercises when they are encountered and before continuing with the text. Additionally, the programs should at least be coded before continuing with the next chapter.

Many students will be able to work through the text without assistance, allowing the tutor to concentrate his attention on those students who are unable to complete the text on their own.

The whole text is designed on the premise that one sure way to learn a programming language is by plenty of practical experience. That is one reason why practical exercises have been inserted throughout the text and a program included at the end of each chapter. The text is also designed so that students can code and run programs at the earliest possible stage in a course.

Only one program is specified at the end of each chapter so that the student can concentrate his effort on that one program. It is important that the student be encouraged to produce a fully correct working version of each program - one which adheres exactly to the program specification. Students should present only correctly working programs for assessment so that the tutor can concentrate on the style and technique of the programs. If necessary, the programs can be replaced by other programs specified by the tutor for specialist courses, although the programs in the text have been carefully chosen to cover as many aspects of computing as possible.

Instead of covering all of a topic at once, most topics are spread over a few chapters and each chapter covers parts of a few topics. This enables students to write more relevant programs earlier in a course and it makes the text more interesting thereby helping a student's concentration. Additionally, because a topic is spread throughout the text, a student has time gradually to assimilate the concepts contained within a topic and to absorb each concept before proceeding to the next.

The text is machine independent and follows the FORTRAN 66 standard. Where the standard allows for interpretation by computer manufacturers, common variations are treated. The COMMON and EQUIVALENCE statements are covered only briefly in the text because of their higher level nature and complexity in being used correctly. It is felt, also, that these two statements and the facilities which they provide are best explained in a 'chalk and talk' situation. The COMPLEX facility is not covered in the book because of its specialist nature.

The FORTRAN 77 standard amendments appear at the end of each chapter. Some computer systems already support some of the amendments. It will take several years for all computer systems to have full FORTRAN 77 standardisation. In the meantime, tutors should inform their students which facilities specified in the text are currently available on the computer system they are using.

One problem encountered by tutors teaching programming languages to a group of students is the wide variation in the students' rates of learning the language - even groups of students with the same background. Consequently, a favoured method of teaching programming languages is by tutor-assisted self-instructional methods, so that students may learn at their own rate and not become bored or lost. This text is particularly suitable for such a course - in fact, part of the text has been used as such for a few years.

It will be necessary for the tutor to introduce the students to the particular computer they will be using and to tell them the procedures for using it. If the computer system provides an equivalent choice between batch and terminal use, students should use the batch initially because, although terminals certainly motivate students, they also encourage very poor programming style and technique. Students can equally well be motivated by a good batch facility.

NOTES FOR STUDENTS

The text is not intended as a beginners' teach-yourself text. It is intended to accompany a course of lectures or be used as the text for a tutor-assisted self-instructional course.

However, it is suitable as a teach-yourself text for students who already have an appreciation of computer programming, or who already know a little FORTRAN. Practising FORTRAN programmers may find the text useful for reference purposes, especially the FORTRAN 77 amendments at the end of each chapter.

There is only one way to learn a programming language thoroughly and that is to have plenty of experience in using it - just like learning a foreign language. Proficiency in FORTRAN is acquired by writing it. This text contains many exercises which call for written answers; it is essential that you do write down the answers. Eager students are sometimes tempted to answer the questions mentally in their impatience to make progress with the subject. Resist this temptation!

All exercises should be attempted as you encounter them and, before continuing with the text. The answers to exercises should be written down and checked with the answers given at the end of the book. Some answers include additional information.

If you are unable to answer a question, read the relevant parts of the text again. If you get stuck, or if there is a point that is not clear, ask your tutor.

At the end of each chapter there is a program specification. The program should be coded, run and tested, preferably before passing on to the next chapter, although this may not be practicable; however, the program should at least be coded before looking at the next chapter. The program should be a fully correct working version which adheres exactly to the specification. A program is no use to anyone if it does not do exactly what is required of it.

The main text adheres to the FORTRAN 66 standard. Amendments specified in the later FORTRAN 77 standard are included at the end of each chapter. The computer system you are using may not support all, or any of the amendments - you should check with your tutor which facilities are available on your computer system.

ACKNOWLEDGEMENTS

My thanks to Arthur Radford, for checking my FORTRAN; Roger Barnes, for checking my English; Peter Leadbetter for checking the final text; Leicester Polytechnic Computer Centre for allowing me to use their computer facilities for the testing and production of the programs in the book; and last, but not least, my wife, Susan, for her support in many ways, particularly in the typing of the draft and final copy of the text.

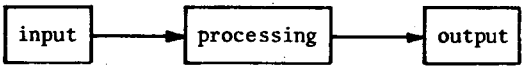
Roger Hutty

Contents

1 *Introduction*

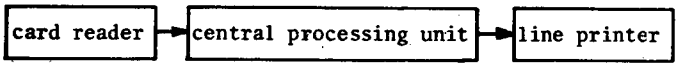
1.1 COMPUTER SYSTEMS

Computing, like many other processes, has three main parts



In a computer system, data (numbers and words) is input by an input device, the processing is performed by a central processing unit and data is output by an output device.

A computer system can have many different types of device but as far as FORTRAN programming is concerned the computer system can be as simple as



A card reader is used to input programs, and data for the programs. A program is a list of statements to tell the computer what to do during the processing stage. A line printer is used to output information, such as results from a program.

Exercise 1.1
Which input device is used for FORTRAN and what is its purpose?

1.2 PROGRAMMING

It takes several stages to get from the specification of a problem to a correctly working program in a computer.

(1) A computer solution of the problem has to be converted to a FORTRAN program and written on a coding form - see Figure 1.1.

(2) The program and data are punched on to cards - see Figure 1.2.

CODING FORM

NAME	STUDENT MARKS	SCHOOL	COMP.	PAGE 1 OF 1 PAGES
USERCODE	R.C. HUTTY	EXT.		DATE

```
      DIMENSION SMARK(5)
      READ(5,10) SMARK
   10 FORMAT(5F5.0)
      SUM=0.0
         DO 20 I=1,5
         SUM=SUM+SMARK(I)
   20    CONTINUE
      AVRAGE=SUM/5.0
      WRITE(6,30) SMARK,AVRAGE
   30 FORMAT(1H ,5F5.1,F8.1)
      STOP
      END
*DATA
 7.5  6.0  8.5  9.0  5.5
```

SM 10
SM 20
SM 30
SM 40
SM 50
SM 60
SM 70
SM 80
SM 90
SM 100
SM 110
SM 120

Figure 1.1

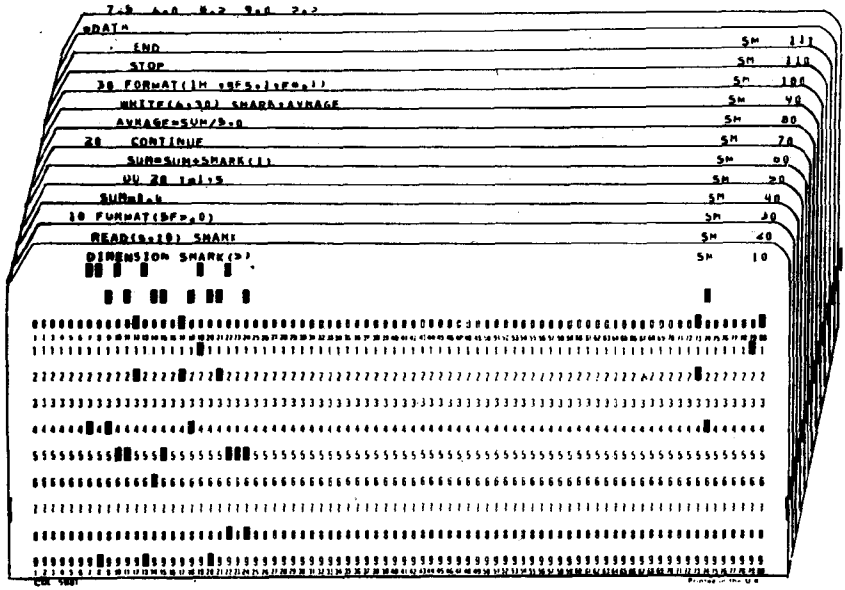


Figure 1.2

(3) The cards are input by the card reader and the program is stored in the central processing unit where it is compiled. During compilation a FORTRAN program is converted into machine code - an equivalent set of instructions, but simple ones, which the computer is able to execute.

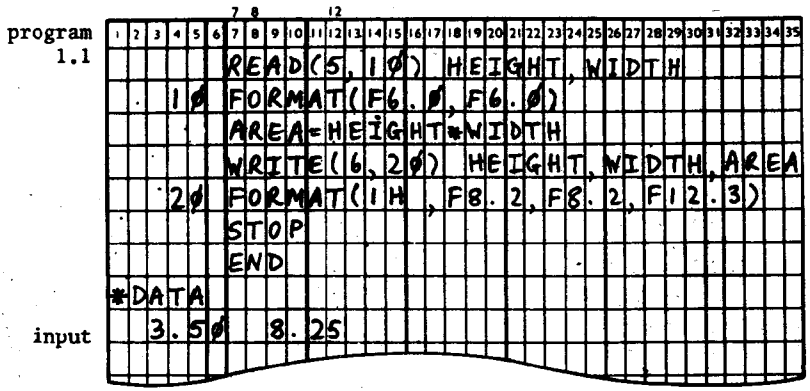
(4) The program is now executed (or run). The data cards will be input by the card reader, the calculation will be performed, and the results will be output to the line printer.

It is quite usual to have to go back a stage or more because of errors. It is rare even for a professional programmer to get through all stages in one go and produce a completely error free program which exactly solves the problem. Such is the preciseness of programming!

Exercise 1.2
Why does a FORTRAN program have to be compiled?

1.3 A SIMPLE FORTRAN PROGRAM

We shall now look at a simple FORTRAN program statement by statement. Program 1.1 calculates the area of a rectangular shape given the height and the width. You will find it helpful to copy the program on to a coding form and have it by your side whilst we go through the program.



The first statement - the READ statement - instructs the computer to input two values from a data card; one value for the variable called HEIGHT and a second value for the variable called WIDTH.

The number 5 in the READ statement informs the computer that the data is to be input from device number five. Each device in a computer system is identified by an unique number. We are assuming, here, that the card reader is device number five - you must use your computer system's card reader number.

The number 10 in the READ statement specifies the number of the FORMAT statement to be used when reading the data. The FORMAT statement contains a field descriptor for each variable in the READ statement. The first field descriptor F6.0 informs the computer that the value of the first variable, HEIGHT, is real (has a decimal point) and is punched in the first six columns of the data card. The second field descriptor F6.0 indicates that the second variable, WIDTH, is also real and is in the next six columns of the card.

After execution of the READ statement HEIGHT will have a value of 3.5 and WIDTH a value of 8.25.

When forming a field descriptor, the number of columns allocated must allow for the maximum possible number of integral and fractional digits plus a decimal point character and a possible sign character.

Exercise 1.3
Write a READ and a FORMAT statement which could be used to input three variables A, B and C. Values for A and B are in the range 0.01 to 9999.99 and the value for C is in the range -99.9 to +99.9.

The assignment statement AREA=HEIGHT*WIDTH instructs the computer to multiply the value of HEIGHT by the value of WIDTH and assign the result to the variable AREA. So after execution of this statement AREA will have the value 28.875.

The WRITE statement instructs the computer to output the values of the three variables HEIGHT, WIDTH and AREA. The number 6 informs the computer that the values are to be output to device number six, which we will assume is the line printer.

The number 20 indicates that the FORMAT statement numbered 20 is to be used with this WRITE statement. The first field descriptor 1Hb instructs the computer to write on the next line of the paper in the line printer. *The letter* b *is used throughout the text to indicate a blank (space).* The next field descriptor F8.2 indicates that the value of the first variable HEIGHT is to be output as a real number with two decimal places in the first eight character positions - similarly for the second field descriptor F8.2 and the variable WIDTH. The last field descriptor F12.3 indicates that the value of the variable AREA is to be output as a real number with three decimal places in the next twelve character positions.

Exercise 1.4
Write a WRITE and a FORMAT statement to output the variables A, B and C of the previous exercise.

The STOP statement stops the execution of the program. A STOP statement need not necessarily be at the end of a FORTRAN program, as you will see later.

The END statement informs the compiler that there are no more FORTRAN statements to be compiled. There is only one END statement in a FORTRAN program, and it must always be the last statement.

The *DATA statement is not a FORTRAN statement. It is a system statement used to separate a program from its data. It varies from one computer system to another - you will have to find out what to put here for the computer system you are using.

The data which is to be input by a program follows the *DATA statement.

When writing a FORTRAN statement on a coding form you should start it in column 7. Statement numbers, for those statements which need one, should be in columns 1 to 5, and right-justified.

1.4 REAL CONSTANTS AND VARIABLES

A real constant is a number with a decimal point - at this stage you must always write a real constant whenever a number is required. No commas are written in real constants and there must be no spaces between the digits.

Look at the following examples

2.0 823.6 0.43 25875.0 0.0

Exercise 1.5
Write the numbers 76 55,760.72 1 and 67½ as real constants.

A variable name consists of from one to six letters or digits, the first of which must be a letter. The first letter of a real variable name must not be any of the letters I,J,K,L,M or N. You must use real variable names at this stage.

Exercise 1.6
Which of the following are valid real variable names?

LENGTH PAGE1 AVERAGE 2ND NUMBER ADRESS

Within the rules given, you are free to choose your own names. However, your choice of names can make a great deal of difference to the intelligibility of your programs. A name should indicate the use of a variable. For example, SALARY=WAGE*HOURS conveys much more information than SA=W*H. Do not use names in the same program which are likely to cause confusion due to their similarity, such as, SUM and SUN.

1.5 ARITHMETIC ASSIGNMENT STATEMENTS

An arithmetic assignment statement is constructed as follows

variable = arithmetic expression

When the statement is executed the arithmetic expression is evaluated and the result is assigned to the variable.

An arithmetic expression consists of variables and operators. The operators available are

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

An example of an arithmetic assignment statement is

```
RESULT=A-B/C*D
```

To human beings this statement would be ambiguous. The arithmetic expression could be interpreted as

$$\frac{a-b}{c \times d} \quad \text{or} \quad a-\left\{\frac{b}{c} \times d\right\} \quad \text{or} \quad a-\frac{b}{c \times d} \quad \text{or} \quad \frac{(a-b)}{c} \times d$$

In FORTRAN, ambiguity is avoided by a few rules governing the order in which operators are evaluated.

First, operators are evaluated in the order

first	**
second	* and /
third	+ and -

Secondly, if an expression contains more than one operator from the same group they are evaluated in order from left to right.

Exercise 1.7
How will the expression in the statement above be interpreted?

Thirdly, parentheses may be used to override the order of evaluation because expressions in parentheses are evaluated first. Expressions within parentheses are evaluated according to the two rules given above. If there are parenthesised expressions within parenthesised expressions then the innermost parenthesised expressions are evaluated first.

Three points to note are: the * operator must always be used when multiplication is required; two operators cannot be next to each other, so that, for example, you must write VALUE**(-2.5); if an exponent is a whole number as, for example, X**2 it may be written without the decimal point.

Exercise 1.8
Write FORTRAN expressions to compute the following

(1) $\left(\frac{x + y}{x + z}\right)^3$ (2) $\frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$ (3) $\frac{(1 + r)^{-p}}{p(p - 1)}$

When a value has been assigned to a variable it may be used later on in a program. Look at Program 1.2.

```
program       READ(5,10) X1,X2,X3
   1.2     10 FORMAT(3F5.0)
              SUM=X1+X2+X3
              SQSUM=SUM*SUM
              WRITE(6,20) X1,X2,X3,SUM,SQSUM
           20 FORMAT(1H ,3F5.1,F8.1,F14.1)
              STOP
              END

  input  23.2 55.7 89.3

 output  23.2 55.7 89.3   168.2        28291.2
```

The sum of the three numbers is assigned to the variable SUM which is then used in the next statement to compute the square of the sum SQSUM. Notice the shorthand way of writing identical consecutive FORMAT field descriptors - 3F5.0 is equivalent to writing F5.0,F5.0,F5.0.

Exercise 1.9
Write a program which inputs the diameters of two concentric circles and outputs the area of each circle and the area between the circles.

PROGRAM

The area of a triangle with sides of length a, b and c can be calculated using the formula

$$\text{area} = \sqrt{(s(s - a)(s - b)(s - c))} \text{ where } s = \frac{a + b + c}{2}$$

Write a program which inputs the lengths of the three sides of a triangle, calculates the area and finally outputs the three lengths and the area.

The three lengths should be on one data card and the program should allow for each length to be in the range 0.1 to 9999.9.