Comparative Study of Computer Languages

Program Organisation (1)

Part XIV

R. Ramaswamy

A program is a set of instructions written in a computer language, for solving a problem. Each instruction is called a statement in the computer language. Statements in a computer language are analogous to sentences in a spoken language. Just as a sentence gives complete meaning to the humans, a statement gives complete meaning to the computer. Just as a story consists of a set of logically organised sentences, a program consists of a set of logically organised statements.

Minimum statements required to program simple problems

What are simple problems? By simple problems we mean problems whose solutions can be obtained by means of a single formula. We have already pointed out that a problem may have either a single formula or a set of formulae. When the problem solution requires the computation of more than one formula, the logic is said to be complex, otherwise the logic is simple or straightforward

A program must read the data and the processing steps. It must print the result on paper. We need statements to do the above functions. First, we need an input statement to read the data. Next, a computation statement to compute the result. Lastly, an output statement for printing the result on paper. Some comment statements inside the program will be useful to the readers to know what the program is about and what it is doing at every stage. Just as a story is organised by logically sequencing the sentences, a program is organised by logically sequencing the different statements. The minimum statements for writing a program are:

1. Input statement
2. Computation statement
3. Output statement
4. Comment statement

We have already seen how the first three types of statements can be written in the different languages. Now we can see how a simple program can be organised in each of the languages. The comment statements will be explained as we describe the program organisation.

FORTRAN program organisation

A program consists of a set of statements, written on a line by line basis. Each line is 80 characters long. In FORTRAN only one statement can be written in a line. There are some restrictions as to the columns in which the statements must be written. Statements can be written only from columns seven to 72. We have said that statements are referenced by giving labels or names in the form of integer numbers.

The statement numbers are written in columns two to five. If the statement is longer than one line, it can be continued in the next line. To tell the computer that a line contains the continuation of the statement from the previous line, a character, say, 1 (or any other character) is keyed in column six. So, the column six is called the continuation mark column. The statement can be continued in the third, fourth, fifth line and so on, every time keying a different character in the sixth column. The computer senses the end of a statement by the absence of any continuation mark in the sixth column of the next line.

Columns 73 to 80 are left blank. The first column is reserved to give indication to the computer that the remaining information keyed in that line are comments intended for the readers and not intended for processing purposes. This indication is given to the computer by keying the letter C in the first column. So, the first column is called the comment column. Due to the restrictions in the columns in which the different items have to be keyed, we say that a FORTRAN program is highly column oriented.

Program files

A program must be stored in a machine decipherable media before it can be used for processing data. In the microcomputer environment, floppy diskettes are used for storing programs as well as data. Just as the humans use paper for storing information, the computer uses the floppy diskettes for storage purposes. The storage media is called the auxiliary memory. It is also called the computer file or, simply, file. When we write the program in the floppy diskette, we say that we store the program as program file. One can ask why not store the program in paper media itself, just as humans do.

Unfortunately, machines cannot decipher what is written on paper. (Hopefully such machines will come to popular use very soon.) Humans cannot decipher what is written on the floppy diskettes. The humans first write the program on paper files. Then they transfer the same to the machine decipherable media—the floppy diskette—using the computer itself or suitable data entry machines. When you store a number of programs in a file, you must be able to call any one program at a time. For this purpose, programs are labelled or given names. A program name is coined as per certain rules. The general form for a program name is

X.L

where L is a three-character code indicating the language in which the program is written, called the secondary name. Since we write the program in FORTRAN, we give the secondary name as FOR. The letter X stands for what is called the primary name coined by combining a maximum of eight characters suggestive of the nature of the program.

Suppose you are writing a program for payroll, you can give the primary name as PAYROL. The primary and the secondary names are separated by a period. The program names in the different languages are:

PAYROL.FOR	(in FORTRAN)
PAYROL.COB	(in COBOL)
PAYROL.BAS	(in BASIC)
PAYROL.PLI	(in PL/I)
PAYROL.PAS	(in PASCAL)

You can access any program in the file by calling its name, just as we access any person by calling his name.

A FORTRAN program for finding simple and compound interest

We will now make use of the statements we have studied so far and see how we can organise a FORTRAN program for finding the simple and compound interest for a principal P, at a rate of R per cent and for a period of N years. Input statements are given to access the data through the console. The program looks as shown in Box I.

Box I

```
C       PROGRAM FOR FINDING SIMPLE & COMPOUND INTEREST
        WRITE(1,20)
 20     FORMAT(1X,'ENTER THE VALUES FOR P,R &N: ')
        READ(1,21) P,R,N
 21     FORMAT(2F7.2, I3)
        SI = P*R*FLOAT(N)/100.0
        CI = P*(1.0+R/100.0)**N - P
        WRITE(1,22)
 22     FORMAT(1X,'PRINCIPAL RATE YEARS S.INTEREST C.INTEREST')
        WRITE(1,23) P,R,N,SI,CI
 23     FORMAT(F10.2, F5.1, I6, 2F11.2)
        STOP
        END


ENTER THE VALUES FOR P,R &N: 100.0,10.0,5

PRINCIPAL RATE YEARS S.INTEREST C.INTEREST
   100.00 10.0     5       50.00       61.05
```

Explanation of the program

The FORTRAN program shown in the Box I has 13 lines. The first statement is a comment statement, since it starts with letter C in the first column. The information meant for the readers is called documentation. By documentation we mean any explanation, verbal or pictorial, given to the readers for the understanding of the problem and the program. The second and the third lines contain the WRITE-FORMAT pair statements for printing a prompt string to tell the programmer about the values to be entered through the console.

The fourth and the fifth lines give the READ-FORMAT pair statements to get the data through the console. The sixth line contains the computation statement for computing the simple interest. In order to convert the integer variable N to real, the library function FLOAT (N) is used. The seventh line contains the computation statement for computing the compound interest. The eighth and the ninth lines contain the WRITE-FORMAT pair statements for printing the headings. The tenth and the eleventh lines contain the WRITE-FORMAT pair statements for printing the results under the proper headings. The twelfth line contains the STOP statement while the thirteenth line contains the END statement. These two statements require further explanation which is given below.

The STOP and the END statements

Any program written in a high-level language is processed in two stages. The first stage is called the compilation stage and the second stage is called the execution stage. During the compilation stage the computer reads the high-level language program, called the source program, and checks up the grammatical correctness of the statements and, if all the statements are correct, it converts the source program into the machine language program called the object program.

We have already pointed out that whatever the language in which you write the program, the computer can execute the program only if it is in the machine language. If there are grammatical errors, the computer will print those errors. The programmer has to correct those errors and again submit the program for compilation. It is only during the compilation stage that the computer uses the END statement. The END statement tells the computer that there are no more FORTRAN statements pertaining to that program. Hence, the END statement will occur at the physical end of the program. The END statement is said to be the logical conclusion of the source program.

Once the compilation is successful, the computer will give a NO ERRORS message, thereby implying that the source program has been converted to the object program or the machine language program. When the computer gives the signal that the compilation is successful, the operator gives the execution command. During the execution stage, the computer executes the statements in the sequence in which they are listed and finally, on encountering the STOP statement, it literally halts the operations after printing the word STOP.

Remember that the END statement does not appear in the object program. The last executable statement in the object program is the STOP statement and we say that the STOP statement is the logical conclusion of the object program. Once the STOP statement is executed, the entire program is erased from the main memory and the control is said to be returned to the operating system. It is the operating system residing in the computer that manages all the programs given by the users. If you want the program to be re-run, you have to give the execution command once again.

It may be noted that there may be more than one STOP statement in a program depending on the problem logic, but there can be only one END statement at the physical end of the program. The five lines at the end of Box 1 show the output. When the computer encounters the READ statement, it halts until the operator enters the data for the variables P, R and N through the console. The computer then prints the headings and the results as shown in the last two lines.

FORTRAN program using the data statement

In the last example, we saw how we can give instructions to the computer to get the data through the console. Suppose you want to enter data in the program itself, it can be done by using the DATA statement. The four statements which occur immediately after the comment statement are replaced by the data initialisation statement or the DATA statement as follows:

```
DATA P, R, N/100.0, 10.0, 5.0/
```

There is no change in the other statements. In this program the data for the input variables P, R and N are initialised in the DATA statement.

One drawback in this method is that, if we want to compute SI and CI for another set of values, we have to change the program. That is, the DATA statement must be rewritten with another set of values and the whole program has to be re-compiled. In the former case there is no re-compilation, but there is only re-execution. When the data are given in the DATA statement, the READ-FORMAT pair statements are removed, otherwise the program looks the same. The output will remain the same as shown in the last two lines of Box 1.

Structure of a FORTRAN program

A FORTRAN program is organised as a sequence of statements each of which is either an instruction or an information to the computer or the reader. The statements which give information to the computer belong to the class of declaration statements and they are non-executable. The statements which give information to the readers are called the comment statements and they are also non-executable. By structure we mean the arrangement of the constituents of the system.

By system we mean an organisation made of interrelated elementary functional units. The smallest logical unit of a FORTRAN program is the statement. A FORTRAN program is organised by a series of statements which are logically related to each other. The structure of a FORTRAN program can be schematically represented as shown below.

Each line represents a statement. The series of lines, one after the other, denote the series of statements written, one after the other. Remember that there is a logic in the sequence. This means that if the positions of the statements are changed, the entire meaning will be changed. Simple FORTRAN programs involving straightforward logic can be organised as above. Later we will discuss how programs involving complex logic can be organised.

Structure of a COBOL program

A COBOL program is made of statements just like FORTRAN. But the statements in a COBOL program are grouped in a structure similar to that of a textbook. Just as a textbook is divided into chapters, and chapters are divided into sections and sections are divided into paragraphs and paragraphs are divided into sentences, a COBOL program is divided into four named divisions and divisions are divided into one or more named sections and sections are divided into one or more named paragraphs and paragraphs are divided into one or more sentences and sentences are divided into one or more statements.

A COBOL program enforces further discipline in the structure by allotting well defined places in the program for giving the different specifications or declarations. It consists of four divisions, each with its own logical position and function in the program. The four divisions are:

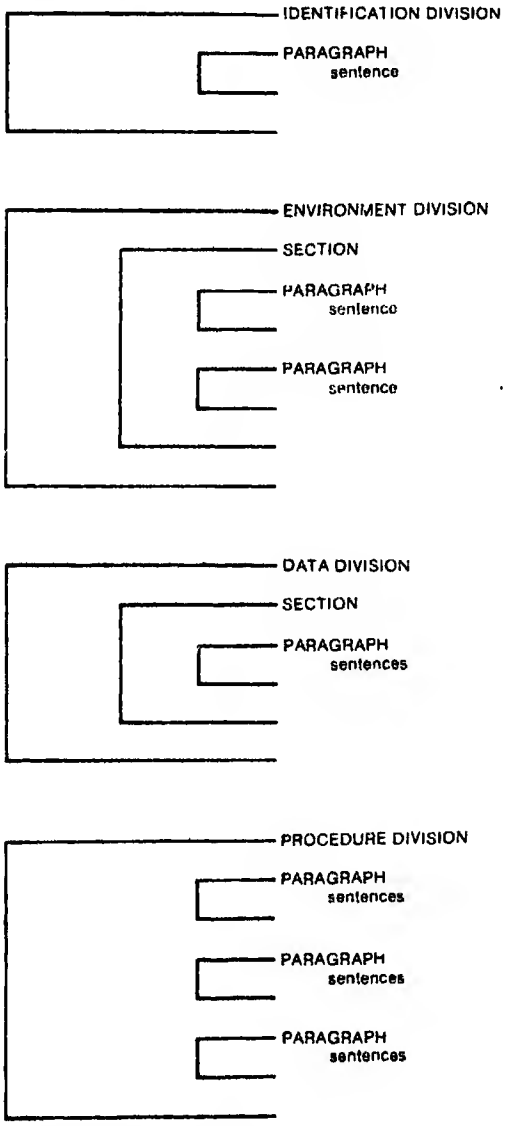
1. IDENTIFICATION division
2. ENVIRONMENT division
3. DATA division
4. PROCEDURE division

Of these four divisions, the actual commands for performing the computations are contained only in the PROCEDURE division. The other three divisions are there only to give information to the computer about the name of the program, the peripheral devices used, the descriptions of data types and so on. Such elaborate formalities are required by COBOL because of the nature of the problem it has to handle.

We know that a computer uses two things, namely the program and the data. Data can be either structured or

unstructured. It is mandatory in COBOL to declare all data, whether they are input or output or generated in the program. The form of declaration for the structured data is different from the instructional data. Now we will consider the structure of a COBOL program which deals only with unstructured data.

A schematic representation of the structure of a COBOL program is shown below:



Rules for key-punching COBOL programs

The COBOL programs are said to be column oriented. This means that different entries must be made from different columns in each line. Each line consists of 80 columns. One character can be punched in each column. There are two margins in the COBOL program and they are called the A-margin and the B-margin. A-margin starts from the eighth column and the B-margin starts from the twelfth column. All divisions, sections and paragraphs have labels or names. These names must be punched from the A-margin. Later we will see that certain level numbers like FD, 01, 77 etc also must be punched from the A-margin.

All sentences and certain level numbers like 88 must be punched from the B-margin. Sentences cannot be keyed beyond column 72. Columns 73 to 80 must be left blank. If one wants to key comment statements, it can be done by putting an asterisk symbol in the seventh column. Every line that contains a comment statement must have an asterisk symbol in the seventh column. Statements can be continued from one line to the next without any restriction, provided you don't break a word. But if a sentence is terminated in one line, a second sentence cannot be keyed from the same line, i.e. more than one sentence cannot be keyed in a line.

It must be noted that statements are separated by blanks and sentences are terminated by periods. The period is a necessary punctuation in COBOL to indicate the termination of sentences, division headers, section headers and para headers. Improper punctuation will give rise to syntax errors.

We will now give some explanations about the entries made in the various divisions. There are difficulties in explaining completely each of these divisions one by one, since in order to understand some entries in one division we must understand their functions in other divisions. This interdependence is somewhat confusing to the beginners, but things will become clear when one gets an overall picture of a COBOL program. Till then the readers should patiently go through the ordeal.

IDENTIFICATION division. This is the first division of a COBOL program. The function of this division is to specify the name of the program by which it will be known to the computer. This division has only one function and so it has only one para. The mandatory entries in the IDENTIFICATION division are

```
A    B
IDENTIFICATION DIVISION.
PROGRAM-ID.
name of the program as coined by the programmer.
```

The first entry is the name of the division, and is identified by the word division and its location. The division header must be punched from the A-margin. The period at the end of the word division is a necessary punctuation. There are no sections in this division. There is only one mandatory paragraph whose name is PROGRAM-ID. The pragraph label must also be punched from the A-margin. The entries must be made from the B-margin.

In this case, the entry is the program name which is coined by the programmer following the same rules for coining data names. A typical name can be PAYROLL. There are some optional paragraphs in this division which are used for documentation purposes. Since these paragraphs are not mandatory, the beginners need not worry much about them. A typical example of entries with the one mandatory paragraph is given below:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
```

The IDENTIFICATION division is the simplest one in the COBOL program and its description is complete, and so we will not come back again to this division since its interdependence with other divisions is little.

ENVIRONMENT division. This is the second division in a COBOL program with two functions, and they are done by two sections. The functions are:

1. To give the names of the computer which compile and execute the program
2. To give the names of peripherals which handle the input and the output files.

And the two sections are:

1. CONFIGURATION section
2. INPUT-OUTPUT section

The CONFIGURATION section is composed of two mandatory paragraphs, namely the SOURCE-COMPUTER, the OBJECT-COMPUTER and one optional paragraph, namely, the SPECIAL-NAMES para. The SPECIAL-NAMES para is used only when one has to connect some system defined names with some programmer coined names. We will see its use later.

The INPUT-OUTPUT section is required only when files are used. Presently we will omit it since we are going to consider independent data items alone. The typical entries in the ENVIRONMENT division of a COBOL which uses only independent data items are shown below:

```
A    B
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. ORION 8000.
OBJECT-COMPUTER. ORION 8000.
```

DATA division. The function of this division is to describe the organisation of the data that will be used in the program. It is composed of two sections, namely the FILE section and the WORKING-STORAGE section. In the FILE section we describe the file data (that is data items present in the input and the output files alone) and in the WORKING-STORAGE section we describe all the independent data items. The typical entries in the DATA division which uses only independent data items are shown below:

```
A    B
DATA DIVISION.
WORKING-STORAGE SECTION.
77 data-name PIC string of A's or X's or 9's
77           "          "
"""""""""""""""""""
"""""""""""""
```

PROCEDURE division. The function of this division is to do all processings with the data and output them in the desired format and in the required devices, either on the printer or on the disc or on the screen. It must be noted that all data used in this division must have been completely described in the DATA division. There is a lot of interdependence between PROCEDURE division and the DATA division. The structure of the PROCEDURE division is shown below:

```
A    B
PROCEDURE DIVISION.
para-header.
   entries.
para-header.
   entries.
""""""""""""""
        """""""""
""""""""""""""
        """""""""
```

In this division we give all the commands to perform the various jobs like reading the data, computing the expressions, printing the results etc.

The commands are given by what are called statements. Statements can be grouped to form a sentence. Sentences can be grouped to form a single logical unit called a paragraph. If necessary, the paragraphs can be grouped to form a section. Normally we do not have sections in the PROCEDURE division. It may be noted that the section headers and the para headers in this division are all programmer-coined.

Summary

COBOL program consists of four divisions, each with its own logical positions and functions. Since these divisions are somewhat interdependent we cannot study these divisions independently one after the other. In other computer languages like FORTRAN or BASIC, one can start writing complete programs after learning to coin statements. But in COBOL one has to study the various formalities required to be entered for writing a complete program.

Now that we have studied the essential statements to write a COBOL program which uses only independent data items and also the formalities to be observed in writing the various statements in the different sections, we will consider a typical COBOL program for finding the simple and the compound interest in the next article. You will find that the statements in the first three divisions are only declaration statements and their function is to give information to the computer about the program and the data which it uses. So we call them non-executable statements, whereas the statements in the last division, PROCEDURE division, are said to be executable statements.

(To be continued)