Comparative Study of Computer Languages

Input Output Statements(4)

Part XIII

R. Ramaswamy

As in the case of FORTRAN, COBOL, BASIC and PL/1, we also have three ways of entering data into a PASCAL program. These are:

1. Entering data through the console.
2. Entering data in the program itself.
3. Entering data in the program by reading a file.

Here we will consider the first two methods and study the third later.

Input statements for entering data through the console

There are two input statements, namely, READ and READLN. The READ statement gets the data given in a stream fashion, while the READLN statement gets the data given on a line by line basis. The general form of the READ statement is

```
READ(list);
```

And the general form of the READLN statement is

```
READLN(list);
```

Suppose we write the READ statement as follows:

```
READ(A,B);
READ(C,D);
```

In the above case the values of A, B, C and D can be keyed in one line or two lines or three lines or even four lines, as shown below

```
34.5 65.7 78.9 87.0          (One line)

34.5 65.7
78.9 87.0                    (Two lines)

34.5 65.7
78.9
87.0                         (Three lines)

34.5
65.7
78.9
87.0                         (Four lines)
```

The first READ statement reads the first two values in the stream. The second READ statement reads the next two values in the stream irrespective of the number of data in each line.

Suppose we give READLN statements as shown below:

```
READLN(A,B);
READLN(C,D);
```

In the above case the values of A and B must be keyed in the first line and the values of C and D must be keyed in the second line. If the values of all the variables are keyed in the same line, the computer will give error message. Both the READ and the READLN statements give instructions to the computer to get the data for the program through the console. The READ and the READLN statements in PASCAL are analogous to the READ statement in FORTRAN, INPUT statement in BASIC, ACCEPT statement in COBOL and GET LIST statement in PL/1.

Output statements for printing the result on paper

There are two output statements to print the result on paper. They are the WRITE and the WRITELN statements. The general form of the WRITE and the WRITELN statements are

```
WRITE(list);
WRITELN(list);
```

The WRITE statement will cause the printing of the values of the variables in the list continuously. If the first line is full the printing will be continued in the second line until the list is exhausted. The WRITELN statement will cause the printing of the values of the variables in the first line. If there are more variables than could be accommodated in the first line, the same will not be printed. Suppose one writes

```
WRITE(A,B);
WRITE(C,D);
```

The values of A, B, C and D will be printed in the same line.

Now if one writes

```
WRITELN(A,B);
WRITELN(C,D);
```

the values of A and B will be printed in the first line and the values of C and D will be printed in the second line. A string in the WRITE or the WRITELN statement will be printed as it is. Numeric quantities will be printed in the exponent notation. Suppose the basic pay of a person is 2500.00 and if one writes a statement

```
WRITELN ('BASICPAY=', BASICPAY);
```

the computer will print the result as

```
BASIC PAY = 2.500000E+03
```

Obviously, the above form is not meaningful. The numeric value must be given in floating point notation. For this purpose we have to give formats for the output.

Formatting the output

When we give a simple WRITE statement or a WRITELN statement with the arguments list inside the parantheses, the computer will print the result in a pre-defined format. But when we want to print the result as per our layout, we have to give necessary specifications explicitly. Such specifications are called format specifications. The field width for each variable can be specified as an integer constant or an expression yielding an integer value immediately after the variable followed by a colon mark. Numbers as well as characters will be printed right justified in the given field. For example, if one writes

```
WRITELN(A:10);
```

and the value of A is 46, the value will be printed in the ninth and the tenth columns, leaving the preceding eight columns blank. If A has been defined as a character variable whose value is 'K', then the letter K will be printed in the tenth column leaving the preceding nine columns blank. If A had been defined as a BOOLEAN variable whose value is TRUE, then the value TRUE will be printed in the columns 7 to 10, leaving the first six columns blank.

The point is that the printing is done always right justified in the given field width, whether the value is a number or a character or a logical constant. If the value of A is 56.78, we can give a specification to print the two digits to the right of the decimal point within a total field width of 10 as follows:

```
WRITELN(A:10:2);
```

The number will be printed in the right justified position in the field width of 10 as

```
bbbbb56.78
```

If you have more than one variable in the WRITELN list, the field specification is given separately for each of the variables. For example, one can write

```
WRITELN (A:10:2, B:7:3, C:5);
```

In the above case, A will be printed right justified in the first ten columns, B will be printed right justified in the second seven columns and C will be printed right justified in the third five columns. There are no other formatting specifications in PASCAL. We find that formatting is the simplest in PASCAL as compared to other languages.

A WRITELN statement without any argument list simply causes the computer to skip to the next line and this is used for giving vertical spacing between lines.

Input statements for entering data in the program

The READ and the READLN statements require the data to be entered through the console. PASCAL has another provision to initialise the data in the program itself. This is done by a declaration statement called the CONSTANT declaration statement. Its general form is as follows:

```
CONST   variable-1 = value;
        variable-2 = value;
        variable-3 = value;
                ....;
                ....;
```

The value associated with the identifier is given as an equation. The value of the identifier must not change in the program by any statement. This means that the variable whose value is declared to be a constant must not change during the program execution.

The reserved word CONST must appear only once, though the values of any number of variables can be defined. The constant values assigned may be either numeric or character or, Boolean or even a string. Constant declarations must be made ahead of the program. The following are examples of valid CONSTANT declarations:

```
CONST   PI = 3.14159,
        TWO = 2,
        BLANK = ' ',
        ALWAYS = TRUE,
        EMPNAME = 'BALACHANDRAN',
        ALPHA = 'A'
```

At the end of the value the punctuation semicolon is a must. In PASCAL, the variables which have been initialised under CONSTANT declaration must not occur on the left-hand side of any assignment statement within the program. This means that these variables cannot be assigned any other value in that particular program. This statement can be used only when constant values are to be initialised and not when changing data values are to be initialised.

Summary of the input statements

Language	General form of statements for entering data through console	General form of statements for entering data in the program
FORTRAN	READ(m,n) list	DATA list/values/
COBOL	ACCEPT variable	77 variable PIC ... VALUE ...
BASIC	INPUT list	READ list
PL/I	GET LIST(list);	DCL(list) type STATIC INIT (values),
PASCAL	READ(list), READLN(list)	CONST variable = value; (only for constant data values)

Thus, we see that all the languages provide facilities for entering data into the program either through the console or entering the data in the program itself. The way of coding the statements are different in the different languages.

Summary of the output statements

Language	General form of output statements for printing the result on paper	Remarks about formats
FORTRAN	WRITE(m,n) list	A separate format statement must be given
COBOL	DISPLAY list	Formats must be given separately
BASIC	PRINT list PRINT USING formats or format variable; list	Formats can be given either in the PRINT statement or separately
PL/I	PUT LIST(list); PUT EDIT(list) (formats or R (format label));	Formats can be given either in the PUT LIST statement or separately
PASCAL	WRITE(list); WRITELN(list);	Formats must be given in the WRITE or the WRITELN statement

This form of output statements will print the result on paper and also display the result on the screen.

Self-testing assignment

You have two variables A and B whose values are 23.7 and 44 respectively. Write the input and the output statements in the proper sequence: (a) to enter the data through and console and print them on paper, and (b) to enter the data in the program itself and print them on paper in all the five languages. Give formats wherever necessary.

FORTRAN

Inputting through console and outputting on paper

```
      INTEGER B
      READ(1,10) A,B
10    FORMAT(F5.1, I3)
      WRITE(2,12)A,B
12    FORMAT(1X,F5.1,I5)
```

Inputting through the program and outputting on paper

```
      INTEGER B
      DATA A,B/23.7,44/
      WRITE(2,12)A,B
12    FORMAT (1X,F5.1,I5)
```

COBOL

```
77  A PIC 999V9.
77  B PIC 999.
77  C PIC ZZZ.9.
77  D PIC ZZZZZ.
    ACCEPT A
    ACCEPT B.
    MOVE A TO C
    MOVE B TO D.
    DISPLAY C, D
```

```
77  A PIC 999V9 VALUE 23.7.
77  B PIC 999 VALUE 44.
77  C PIC ZZZ.9
77  D PIC ZZZZZ.
    MOVE A TO C.
    MOVE B TO D
    DISPLAY C, D
```

BASIC

```
INPUT A, B
PRINT USING "### # ####";
 A,B
```

```
READ A, B
DATA 23.7, 44
PRINT USING "###.# ####",
 A,B
```

PL/1

```
DCL A FLOAT BINARY,
    B FIXED BINARY;
GET LIST (A, B),
PUT SKIP EDIT(A, B)
 (F(5,1),F(5));
```

```
DCL A FLOAT BINARY
 STATIC INIT (23.7),
 B FIXED BINARY
 STATIC INIT(44);
PUT SKIP EDIT(A,B)
 (F(5,1),F(5));
```

PASCAL

```
VAR A, REAL; B, INTEGER;
READLN(A,B),
WRITELN(A:5:1, B:5);
```

```
CONST A = 23.7; B = 44;
WRITELN(A:5:1, B:5);
```

It may be noticed that only in BASIC no declarations are made about the variable types. In FORTRAN declaration is made only for the variable B, since the type has to over-ride the predefined convention. There is no declaration for A since its value is according to the predefined convention.

In all the above cases the result will be printed in the same way as shown below:

```
23.7  44
```

What about string data items?

Only a string of maximum four characters length can be stored in a single FORTRAN variable. In PASCAL, only

one character can be stored in a variable and the variable is called the character variable. Strings can be read, assigned, manipulated and printed with facility only in the three languages, COBOL, PL/I and BASIC.

Another self-testing assignment

A string constant 'KRISHNAN' is to be stored. Write the input and the output statements in the proper sequence: (a) to enter the data through the console and print them on paper, and (b) to enter the data in the program itself and print them on paper in all the five languages. Give the declaration statements and formats wherever necessary.

FORTRAN

Inputting through console and outputting on paper	Inputting through the program and outputting on paper
READ (1,10) A,B 10 FORMAT(2A4) WRITE(2,12)A,B 12 FORMAT(1X, 2A4)	DATA A,B,'KRIS', 'HNAN'/ WRITE(2,12) A,B 12 FORMAT(1X, 2A4)

COBOL

77 A PIC A(8). ACCEPT A. DISPLAY A.	77 A PIC A(8) VALUE 'KRISHNAN' DISPLAY A

BASIC

INPUT A$ PRINT A$	READ A$ DATA "KRISHNAN" PRINT A$

PL/1

DCL A CHARACTER(8); GET LIST(A); PUT LIST(A);	DCL A CHARACTER(8) STATIC INIT ('KRISHNAN'), PUT LIST(A);

PASCAL

VAR A,B,C,D,E,F,G,H: CHAR. READLN(A,B,C,D,E,F,G,H); WRITELN(A,B,C,D,E,F,G,H).	CONST A = 'KRISHNAN', WRITELN(A),

In FORTRAN, we have stored the name 'KRISHNAN' in two variables A and B since one variable can read and write only four characters. The variables are read and written in the A format. There are no declaration statements for that in FORTRAN.

In COBOL there is no problem, since a variable which is declared as belonging to alphabetic data type can store long strings.

In BASIC, we have a different way of coding string variables. A dollar sign at the end of the variable indicates that it is a string variable and it can store long strings.

In PL/1, a string variable is declared as CHARACTER with its length specified in the bracket.

In PASCAL, a single variable can read and write only one character. Since the string data has eight characters, eight variables are declared. Then they are read and written. When a PASCAL variable is declared as a CONSTANT with a value, it can store a string. So we have given a CONSTANT declaration giving the value of the variable A equal to the string data. Then the constant variable is given in the WRITELN statement.

When we come to array structure for data we will see how the string data can be more efficiently handled in FORTRAN and PASCAL.

Some points to remember

The input to the computer is raw data. The output from the computer is processed data. The computer makes use of the program and converts the raw data into a meaningful data. The computer is simply a data converter and we may call it a data processing machine.

Remember that all processings are done completely automatically by using the stored instructions or the program. That is, the computer can store both the program and the data in its memory. Once it is started, it can do all the operations on the data as per the program of instructions completely automatically without any human intervention at any stage of operation and output the result.

It is this automaticity coupled with the fantastic speed that has made the computer so versatile. The computer is a superb marvel of modern technology. It is progressing by leaps and bounds, and we can expect many more marvels in the years to come. Man is constantly generating new knowledge and one has to be a learner throughout life!

(To be continued)