12

INPUT AND OUTPUT OF DATA

12.1 INTRODUCTION

In 'C' language input and output of data is done by a collection of library functions like getchar, putchar, scanf, printf, gets and puts. These functions permit the transfer of information between the computer and the standard input/output devices. The library function getchar and putchar as the name suggests, allow single characters to be transferred into and out of the computer, scanf and printf permit the transfer of single characters, numerical values and strings, gets and puts facilitate the input and output of strings. An input/output function can be accessed from anywhere within a program simply by writing the function name, followed by a list of parameters enclosed in parentheses. Some input/output functions do not require parameters, but the empty parentheses must appear. 'C' include a collection of header file that provide necessary information in support of the various library functions. The header file stdio.h contains the information about input/output library functions. In this lesson we will discuss some input/output functions in detail.

12.2 OBJECTIVES

After going through this lesson you would be able to

- explain getchar function
- define putchar function

- explain scanf function
- explain printf function
- describe gets & puts function
- use interactive programming

12.3 GETCHAR FUNCTION

getchar function reads a single character from standard input. It takes no parameters and its returned value is the input character.

In general, a reference to the getchar function is written as character variable = getchar();

For example char c;

c= getchar () ;

The second line causes a single character to be entered from the standard input device and then assigned to c.

If an end-of-file condition is encountered when reading a character with the getchar function, the value of the symbolic constant EOF will automatically be returned.

This function can also be used to read multicharacter strings, by reading one character at a time within a multipass loop.

12.4 PUTCHAR FUNCTION

The standard C function that prints or displays a single character by sending it to standard output is called putchar. This function takes one argument, which is the character to be sent. It also returns this character as its result. If an error occurs, an error value is returned. Therefore, if the returned value of putchar is used, it should be declared as a function returning an int.

For example

```
putchar ('N');
putchar ('a');
putchar ('t');
putchar ('i');
putchar ('o');
putchar ('n');
putchar ('a');
putchar ('l');
```

When putchar is used, however, each character must be output separately. The parameter to the function calls in the given statements are character constants, represented between apostrophes as usual. Of course, the arguments could be character variables instead.

Two functions that require FILE pointers are getc and putc. These functions are similar to getchar and putchar, except that they can operate on files other than the standard input and output. The getc function is called with one argument, which is a FILE pointer representing the file from which the input is to taken. The expression getc(stdin) is similar to

getchar()

and the expression putc(c, stdout) is same as putchar(c).

INTEXT QUESTIONS

1. What is the difference between input and output ?
2. What are the restrictions related to the putchar function ?
3. Name the counterpart to the putchar function.

12.5 SCANF FUNCTION

Input data can be entered into the computer from a standard input device by means of C library function scanf. This function can be used to enter any combination of numerical values, characters single character and strings. The function returns the number of data items that have been entered successfully.

In general terms, the scanf function is written as

scanf (string, parameter 1, parameter 2..., parameter n);

Where string= string containing certain required formatting information, and Parameter 1, parameter 2.. = parameters that represent the individual input data item.

The control string or string comprises individual groups of characters, with one character group for each input data item. Each character group must start with percent sign (%). In the string, multiple character groups can be contiguous, or separated by white space characters. The conversion character that is used with % sign are

many in number and all have different meaning corresponding to type of data item that is to be input from keyboard.

Some of the conversion characters are listed below:-

Conversion character	Meaning
c	type of data item is single character
d	type of data item is decimal integer
e	type of data item is floating-point value
f	type of data item is floating-point value
h	type of data item is short-integer
i	type of data item is decimal, hexadecimalor octal integer
o	type of data item is octal integer
s	type of data item is string
u	type of data item is unsigned decimal integer
[. . .]	type of data item is string which may include whitespace characters.

The parameters are written as variables or arrays, whose types match the corresponding characters groups in the control string. Each variable name must be preceded by an ampersand (&).

For example: Suppose there are 3 variables char name[10], int roll_no, float marks, then the scanf statement for these 3 variables will be

scanf("%s %d %f," name, &roll_no, &marks);

This statement contains three character groups. %s represents that first parameter is string i.e name. The second character group %d, represents that the parameter is decimal integer value, and third character group %f represents that the parameter is floating point value.

It is to be noted here that roll_no and marks that are only variables not arrays and are preceded by ampersand sign unlike name which is a character array. The order of data items depends upon the requirement of user, but the order of character group should match with the order of data items.

If two or more data items are entered, they must be separated by white space characters. Data items may continue onto two or more lines, since the newline character is considered to be a whitespace character.

It is already stated that s-type conversion character applies to a string terminated by a whitespace character. Therefore, a string that includes whitespace characters cannot be entered in this manner. To do so the s-type conversion character within the control string is replaced by a sequence of characters enclosed in square brackets designated as [...]. Whitespace characters may be included within the brackets, thus accommodating strings that contain such characters.

When the program is executed, successive characters will continue to be read from the standard input device as long as each input character matches one of the characters enclosed within the brackets. The order of the characters within the square brackets need not correspond to the order of the characters being entered. Input characters may be repeated. The string will terminate, however once an input character is encountered that does not match any of the characters within the brackets. A null character \0 will then automatically be added to the end of the string.

Another way to do this is to precede the characters within the square brackets by a circumflex (^). This causes the subsequent characters within the brackets to be interpreted in the opposite manner. Thus, When the program is executed, successive characters will continue to be read from the standard input device as long as each input character does not match one of the characters enclosed within the brackets. If the characters within the brackets are simply the circumflex followed by a **new line** characters, then the string entered from the standard input device can contain any ASCII characters except the newline character. For example:

```
char school [40];

scanf  ("%[^\n]", school);
```

Through this statement any string of undetermined length (but not more than 40 characters) will be entered from the standard input device and assigned to school.

"National Open School" is assigned to array **'school'**. If you want to limit or to restrict the width of the data item you can define it with the help of an unsigned integer indicating the field width is placed

within the control string, between the % and the conversion character.

The data item may be composed of fewer characters than the specified field width. But you cannot exceed the number of characters in the actual data item than specified field width. Any character that extend beyond the specified field width will not be read, incorrectly interpreted as the components of next data item. Such leftover characters may be

For example:

```
int x,y,z;
scanf("%3d % 3d %3d", & x, &y, &z);
```

If the data input from the keyboard is 1,2,3 then the result is

x=1, y=2 , z=3 but suppose if the data input is 123 , 456, 789 then it will result in

x=123, y=456, z=789. If the input is 1234 5678 9 then x=123 y=4 z=567.

The remaining two digits (8 and 9) would be ignored, unless they were read by a subsequent scanf statement.

If the control string contains multiple character groups without interspersed whitespace characters, then whitespace characters within the input data will be interpreted as a data item. To skip this and read the next nonwhitespace character, the conversion group %s should be used.

INTEXT QUESTIONS

4. What function enables a user to input information while the program is in execution ?

5. If numeric or single-character information is being entered by means of scanf function, what symbol must precede the corresponding variable name ?

12.6 PRINTF FUNCTION

The printf function is used to print out a message, either on screen or paper(The letter "f" in printf could stand for either "formatted" or

"function"). It is equivalent to the WRITE statement in Pascal, only more powerful. It is similar to the input function scanf except that its purpose is to display data rather than to enter data into the computer. So, printf function moves data from the computer's memory to the standard output device, whereas the scanf function enters data from the standard input device and stores it in the computer's memory. The general form is:

printf(string, parameter1, parameter2,……, parameter n)

where string refers to a string that contains formatting information, and parameter 1, parameter2… parameter n are arguments that represents the individual output data items. The parameters can be written as constants, single variable or array names or more complex expressions.

Unlike scanf function, the parameters in a printf function do not represent memory addresses and therefore they are not preceded by ampersand (&) sign. The control string or string is composed of individual groups of characters, with one character group for each output data item. Each character group must start with a percent sign like in scanf function followed by a conversion character indicating the type of the corresponding data item. Multiple character groups can be contiguous, or they can be separated by other characters, including whitespace characters. Some of the conversion characters are given below:

c type of data item is displayed as a single character

d type of data item is displayed signed decimal integer

e type of data item is displayed as a floating-point value with an exponent

f type of data item is displayed as a floating-point value without an exponent

o type of data item is displayed as an octal integer without a leading zero.

s type of data item is displayed as a string

u type of data item is displayed as an unsigned decimal integer

x type of data item is displayed as a hexadecimal integer, without the leading ϕx.

Let us consider an example of printf.

Suppose there are 3 variables, char name[10], int roll_no, float marks,

then the printf statement to display all these 3 variables is printf("%s %d %f"' name, roll_no, marks);

Within the printf function, the control string or string is "%s %d %f". It contains 3 characters groups. The first character group %s indicates that the first parameter represents a string. The second character group, %d indicates that the second parameter represents a decimal integer value and the third character group %f indicates that the third parameter represents a floating point value.

It is to be noted here that the parameters are not preceded by ampersands as in scanf. The printf function interprets s-type conversion differently than the scanf function. In the printf function,s-type conversion is used to output a string terminated by the null character (\0). Whitespace characters may be included within the string.

For example

```
char school[40];
scanf("%[^\n]" ,school);
printf("%s",school);
```

If the string entered through keyboard is "National Open School" then it will print it as it is on the screen through printf statement.

A concept of minimum field width in a printf statement, can be specified by preceding the conversion character by an unsigned integer. If the number of characters in the corresponding data item is less than the specified field width, then the leading blanks will be added in front of data item to fill the specified field. But if the number of characters in the corresponding data item is greater than the specified field width then the extra space will be allocated to display the whole data item as contrary to scanf function in which there is a concept of maximum field width as stated earlier.

It is also possible to specify the maximum number of decimal places for a floating point value or the maximum number of characters for a string. This is known as **precision.** This is an unsigned integer that is always preceded by a decimal point. If a minimum field width is specified in addition to the precision then the precision specification follows the field width specification.

For example

```
float a= 456.789;
printf("%7f % 7.3f %7.1 f", a,a,a);
```

The result is 456.789000 456.789 456.7

Minimum field width and precision specification can be applied to character data as well as numerical data. When applied to a string, the minimum field width is interpreted in the same manner as with a numerical quantity. The field width specification will not prevent the entire string from being displayed. The precision specification will determine the maximum number of characters that can be displayed. If the precision specification is less than the total number of characters in the string, the excess rightmost characters will not be displayed. This will occur even if the minimum field width is larger than the entire string, resulting in the addition of leading blanks to the truncated string.

For example

```
char school[9];

printf("%7s %12s % 12.3s  %.3s, school , school,
school,school);
```

Suppose the string is 'National' then the output is

```
Nationa ....  National..... Nat Nat
```

The first string is shown as input even this consists of 8 characters and the field width specification is 7 characters. So it over rides the minimum field width specification. The second string is padded with 4 leading blanks to make 12 as field width specification. Third string is padded with 12 leading blanks and only 3 characters are shown because of 3 character precision specification and the last string also displays only 3 characters but there is no leading blank spaces because of no minimum field width specification. So second string in this case is right-justified in the output.

In addition to the field width, the precision and the conversion character, each character group within the control string can include a flag, which affects the appearance of the output. The flag must be placed immediately after the percent sign. Some of the flags are given below with their common usage.

1. – data item is left-justified within the field

```
int i=346
printf(":%ϕ-6d",i);
output :346
```

2. + a sign (either + or-) will precede each signed int i=346
float f=-3.5
numerical data item; without this flag, only negative data should be there–3. 5e+000: items are preceded by a sign.
printf(":%-+6d%-10.1e":,i,j);
output is:+346 some space

3. Zero causes leading zeros to appear instead of leading blanks, applies only to data items that are right justified.Within a field whose minimum size is larger than the data item.

4. ' ' a blank space will precede each +ve signed numerical data item; flag is override by the + flag if both are present.

5. # causes octal and hexadecimal data items to be preceded by Zero() and x respectively.

In case of strings there is also a usage of flags. Let us see it with the help of examples:

Suppose string is " National Open"

1	printf(":%s:\n", "National Open");	:National Open:	Normal Printing
2	printf(":%9s:\n", "National Open");	:National Open:	Min.field width
3	printf(":%17s:\n", "National Open");	....National Open:	printed in a field wider than the string.
4	printf(":%-17s:\n", "National Open");	:National Open.... ;	left justified printed in a field of 17.
5	printf(":%17.6s:\n", "National Open");	:..........Nation	truncated to 6 characters
6	printf(":%.6s:\n", "National Open");	: Nation:	: truncated to 6 characters
7	printf(":%-17.6s:\n", "National Open");	: Nation.....:	truncated to 6 characters left justified in a field of 12.

Unrecognised characters within the control string or string will be displayed just as they appear. This feature allows us to include label and messages with the output data items if we want.

There is some variation in the features supported by the printf function in different versions of C.

INTEXT QUESTIONS

6. What conversion specification are used to print out the following integer, a floating point number in decimal & scientific notation & a single character ?

7. What is the minimum field width specifics, & where is it located ?

8. When the conversion specification is %, what is its default field width ?

12.7 GETS AND PUTS FUNCTION

'C' contains a number of other library functions that permit some form of data transfer into or out of the computer. Gets and puts functions facilitate the transfer of strings between the computer and the standard input/output devices. Each of these functions accepts a single argument or parameter. The parameter must be a data item that represents a string. The string may include whitespace characters. In the case of gets, the string will be entered from the keyboard and will terminate with a newline characater.

The gets and puts functions are alternative use for scanf and printf for reading and displaying strings.

For example:

```
char school[40];
gets(school);
puts(school);
```

These lines uses the gets and puts to transfer the line of text into and out of the computer. When this program is executed, it will give the same result as that with scanf and printf function for input and output of given variable or array.

12.8 INTERACTIVE PROGRAMMING

Interactive programming means to create an interactive dialog between user and computer. This is some sort of question and answer. This can be created by alternate use of scanf and printf functions.

This type of interactive programming is useful in the case of useful reports or data entry.

12.9 WHAT YOU HAVE LEARNT

In this lesson you have learnt about getchar and putchar function in detail. You are now aware of scanf, printf functions which are used for input of data from the keyboard and display of data to the screen in a formatted way. You have also learnt the concept of gets and puts which are equivalent to scanf and printf except for formatting.

You can do interactive programming for your project work very easily after going through this lesson.

12.10 TERMINAL QUESTIONS

1. What are the commonly used input/output functions in C? How are they accessed ?
2. What is the standard input/output header file called in most versions of C?
3. Explain the purpose of the getchar function.
4. Define the purpose of scanf function.
5. Summarize the meaning of the more commonly used conversion characters within the control string of a scanf function.
6. What is the purpose of the printf function? How is it used within a C program?
7. Explain the precision of an output data item.
8. Summarize the purpose of the flags commonly used within printf.
9. How are unrecognized characters within the control strings of a printf function interpreted ?
10. Explain the use of the gets and puts function ?

12.11 KEY TO INTEXT QUESTIONS

1. Information that enters the computer from the outside is called input, and information produced by the computer and sent to the outside is known as output.

2 It can outputs only one character at a time, and only to the standard output.

3 getchar function

4. scanf function

5. The ampersand (&) symbol.

6. Integer %d, floating point in decimal %f, with scientific notation %e, a single character % c.

7. It specifies the smallest field in which a value is to be printed. It is used in a conversion specification, and if present, it is placed immediately following the % sign.

8. The size of the number being printed (including a minus sign, if the number is negative.)