RV College of Engineering®

Go, change the world

PROGRAMMING IN C (22CS23)

UNIT-I
Handling Input Output

C provides several functions that give different levels of input and output capability.

The input and output functions in C are built around the concept of a set of standard data streams being connected from each executing program to the basic input/output devices. These standard data streams or files are opened by the operating system and are available to every C and assembler program for use without having to open or close the files.

These standard files or streams are called

- stdin : connected to the keyboard
- stdout : connected to the screen
- stderr : connected to the screen

The input / output functions fall into two categories:

1. **Unformatted read (input) and display (output) functions**

   getchar() and putchar()

1. **Formatted read (input) and display (output) functions**

   scanf() and printf()

UNFORMATTED INPUT AND OUTPUT

Reading a character - The getchar( ) function:

Single characters can be entered into the computer using the C library function getchar( ). The getchar( ) function is a part of the standard C I/O library. It reads a single character from the standard input device, typically a keyboard. The function does not require any arguments, though a pair of empty parentheses must follow the word getchar.

The general syntax

Character variable=getchar();

Where character variable is the name of a variable of type char.

The getchar() function receives the character data entered, through the keyboard, and places it in the memory location allotted to the variable **char_variable**.

Writing a character - The putchar( ) function:

Single characters can be displayed using the C library function putchar(). This function is complementary to the character input function getchar. The putchar function, like getchar, is a part of the standard C I/O library.. It must be expressed as an argument to the function, enclosed in parentheses, following the word putchar.

Syntax: putchar(char_variable);
where char_variable is the name of a variable that is of type

char. It transmits a single character to a standard output device.

Example:

```
#include<stdio.h>

int main(void)

{
    int ch;
    ch=getchar();
    putchar(ch);
    return 0;

}
```

Input: A

Output: A

Additional Single Character Input and Output Functions:

Other than getchar() and putchar(), there are some more single character input and output functions that are available in Turbo C Only which are listed below:

- getch() – This input function reads, without echoing on the screen, a single character from the keyboard and immediately returns that character to the program.

General statement form: ch = getch();

getche() This input function reads, with echo on the screen, a single character from the keyboard and immediately returns that character to the program.

General statement form: ch = getche();

Additional Single Character Input and Output Functions:

putch() This output function writes the character directly to the screen. On success, the function putch() returns the character printed. On error, it returns EOF.

General statement form: putch(ch);

```
#include<stdio.h>

int main( )

{

        int ch;
        printf(“\n continue(Y/N) ?”);

        ch = getch( );

        putch(ch);

        return 0;

}
```

Result: continue(Y/N) ? Y

```
#include<stdio.h>

int main( )

{

        int ch;
        printf("\n continue(Y/N) ?");

        ch = getche( );

        return 0;

}
```

Result: continue(Y/N) ? N

FORMATTED INPUT AND OUTPUT

Input Function scanf () :

The function scanf() is used to read data into variables from the standard input, namely a keyboard. The general format is:

scanf(format-string, var1,var2,……varn)

Where format-string gives information to the computer on the type of data to be stored in the list of variables var1,var2……varn and in how many columns they will be found.

For example, in the statement:

scanf("%d %d", &p, &q);

The two variables in which numbers are used to be stored are p and q. The data to be stored are integers. The integers will be separated by a blank in the data typed on the keyboard.

The scanf statement causes data to be read from one or more lines till numbers are stored in all the specified variable names. The symbol & is very essential in front of the variable name.

If some of the variables in the list of variables in the list of variables in scanf are of type integer and some are float, appropriate descriptions should be used in the format-string.

For example:

scanf("%d %f %e", &a , &b, &c);

Specifies that an integer is to be stored in a, float is to be stored in b and a float written using the exponent format in c. The appropriate sample data line is:

485 498.762 6.845e-12

Output Function printf () function:

The general format of an output function is

printf(format-string, var1,var2…..varn);

Where format-string gives information on how many variables to expect, what type of arguments they are , how many columns are to be reserved for displaying them and any character string to be printed. The printf() function may sometimes display only a message and not any variable value. In the following example:

printf(“Answers are given below”);

printf(“Answers are given below\n”); - In this string the symbol \n commands that the cursor should advance to the beginning of the next line.

In the following example:

printf("Answer x= %d \n", x);

%d specifies how the value of x is to be displayed. It indicates the x is to be displayed as a decimal integer. The variable x is of type int. %d is called the conversion specification and d the conversion character . In the example:

printf("a= %d, b=%f\n", a, b);

the variable a is of type int and b of type float or double. % d specifies that a is to be displayed as an integer and %f specifies that, b is to be displayed as a decimal fraction. In this example %d and %f are conversion specifications and d, f are conversion characters.

Example Program

Example to indicate how printf () displays answers.

```
/*Program illustrating
printf()*/  # include<stdio.h>

main()
{
        int a= 45, b= 67
        float x=45.78 , y=34.90
        printf("Output:\n");
        printf("1,2,3,4,5,6,7,8,0\n");

        printf("\n");
        printf("%d, %d, %f ,%f \n" , a,b,x,y);
        printf("\n");

}
```

Output:

- 1234567890
- 45,67,45.78,34.90

Example for illustrating scanf and printf statements:

```
/* Program for illustrating use of scanf and printf statements
*/  #include<stdio.h>

main()
{
        int a,b,c,d;
        float x,y,z,p;
        scanf("%d %o %x %u", &a, &b ,&c ,&d);
        printf("the first four data displayed\n");
        printf(("%d %o %x %u \n", a,b,c,d);
        scanf("%f %e %e %f", &x, &y, &z, &p);
        printf("Display of the rest of the data read\n");
        printf("%f %e %e %f\n", x,y,z,p);
```

Input:

-768 0362 abf6 3856 -26.68 2.8e-3 1.256e22 6.856

Output:

The first four data displayed

-768 362 abf6 3856

Display of the rest of the data read

-26.680000 2.800000 e-03 1.256000e+22 6.866000

End of display