# Input and Output using C programming:

**Introduction →** Any program written has three important states namely, accepting input, processing them using functions and returning or displaying the result as output. Accepting data is done by the keyboard and displaying output is done by the monitor.

C language provides various functions to control input and output of a program. One of them is to use assignment statement using = sign, other method is the use of scanf function.

The same mechanisms can be used to read or write data from and to files. It will be discussed in File input/output chapters.

## The Standard Input Output File →

C language provides a standard library for performing input and output to files or the terminal. Most of the input output functions will be discussed now in this section. This library is known as Standard Input-Output header file represented by the filename 'stdio.h'. Each program that uses these functions, must include the statement — `#include <stdio.h>` at the beginning.

## Character Input/output →

This is the lowest level of input and output. It provides very precise control, but is usually too fiddly to be useful. Most computers perform buffering of input and output. This means that they will not start reading any input until the return key is pressed, and they will not print characters on the terminal until there is a whole line to be printed. Reading or writing a single character can be done by the functions:-

getchar()
putchar()

**getchar()** — returns the next character of keyboard when a key is pressed. Normally the getchar() function is used on the right hand side of an assignment statement, whereby it assigns the character to the variable on the left hand side of = sign.

While reading from a file if it reaches the end of whatever file it is reading, it returns the character represented by '\0' (ASCII NULL, which has value zero).

Example:-

```
#include <stdio.h>
int main() {
        int i=0;
        char ch;
while ((ch = getchar()) != EOF)
  {
    i++;
    printf("%c \t",ch);
  }
printf("\n You have total %d characters", i);
return 0;
}
```

---

Compile & Run this program you will get the same characters as your input and finally counts the no. of characters entered totally.

**putchar()** — puts one character on the standard output each time it is called. putchar() puts its character argument on the standard output (screen).

Following example program converts any typed input into capital letters. To do this we have to use toupper() function defined

in ctype.h header file. But before conversion to upper it checks whether an alphabet has been typed or not. For this we have to use isalpha() function for confiomation about typed character is not a digit, but it is an alphabet. isalpha() function takes a character argument and returns a non-zero (true) value if the argument contains an alphabet; otherwise it assumes a value zero (false).

Example:-

```
#include <ctype.h>
#include <stdio.h>
int main(){
    char ch;
    while ((ch=getchar()) != EOF)
    {   if ( (isalpha(ch) > 0))
        {   putchar (toupper(ch));
        }
    }
return 0;
}
```

Formatted Input/Output → Formatted input refers to an input data that has been arranged in a particular format. Such data has to be read conforming to the format of its appearance. C language provides us with a set of functions to achieve this. We have met these functions earlier in the course. C language has following two formatted functions —

printf()
scanf()

prototype:- printf("control string", arg1, arg2, ... argn);

Some control strings are:-

%d — int
%f — float or double
%e — float or double in scientific notation
%c — character
%s — character string (char *)

It is also possible to insert numbers into the control string to control field widths for values to be displayed. For example %.6d would print a decimal value in a 6 space wide; %.8.2f would print a real value in a field 8 space wide with 2 decimal places (inside). Display is left justified be default, but can be right justified by putting - sign before the format information, for ex:- %-6d, a decimal integer right justified in a 6 space field.

scanf() —

prototype:- scanf ("control string", arg1, arg2, ... argn);

Example:-

```
#include <stdio.h>
int main() {
        float radius;
  char name[15]; int num;
printf ("Enter the radius of circle:");
scanf("%f", &radius);
printf("A circle of radius %f has area %f\n",
                    radius, 3.14 *radius *radius);
printf ("Enter a number from 1 to 1000:");
scanf("%d", &num);
printf ("Your Number is %d \n", num);
printf ("Enter your name:");
scanf ("%s", name);
 printf ("Hello %s \n", name);
return (0);
}
```

In next section we will discuss about whole lines of Input and Output functions —

- gets()
- puts()