Unit-3
Input and Output

Data input and output

- A program without any input or output has no meaning.
- Reading the data from input devices and displaying the result are the two main task of any program.

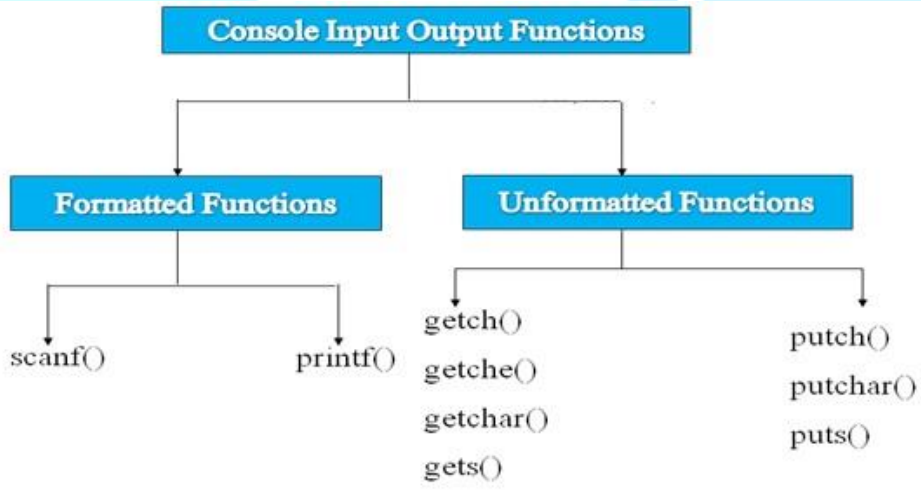
Input:
- It is a process of transferring data from input devices into program.
- C provides a set of built-in functions to read given input and feed it to the program as per requirement.

Output:
- It is a process of displaying data on screen, printer or in any file.
- C provides a set of built-in functions to output required data.

Input and output functions

- Input/output functions are the links between the user and the terminal.
- **Input functions** are used to read data from keyboard are called standard input functions.
  scanf(), getchar(), getche(), getch(), gets() etc.
- **Output functions** are used to display the result on the screen are called standard output functions.
  printf(), putchar(), putch(), puts() etc.
- In C, the standard library **stdio.h** provides functions for input and ouput.
- The instruction **#include<stdio.h>** tells the compiler to search for a file named **stdio.h** and places its contents at this point in the program.
- The contents of the header file become part of the source code when it is compiled.

The input/output functions are classified as follows:
1. Formatted functions
2. Unformatted functions



Formatted Functions

- Formatted functions allow the input from the keyboard or the output displayed on screen to be formatted according to our requirements.
- Input function: scanf( )
- Output function: printf( )

} Formatted functions

Formatted Input

- The well-known function for formatted input is **scanf**.
- The built-in function **scanf()** can be used to enter input data into the computer from a standard input device.
- Its general form is as follows:

  scanf("control string", arg1, arg2,…..,argn);

  Where, control string → format in which data is to be entered.

  arg1, arg2,… → location where the data is stored and preceded by ampersand (&)
- The control string consists of individual groups of data formats, with one group for each input data item.
- Each data format must begin with a percentage sign.

Conversion Specifier:

Conversion character	Description	Example of codes
%d	For an integer in decimal system	`int m = 60; printf ("%d",m);`
%f	For a float-type floating point decimal Number	`floaty= 8.5 ; printf( "%f", y);`
%lf	For double-type floating point decimal Number	`double P = 5.435; printf("%lf",P)`
%c	For a character	`char ch = 'H'; printf("-%c", ch);`
%s	For a string of characters	`char Str[6]= "Thakur; printf("%s",Str);`

E.g.

```
#include<stdio.h>
void main()
{
   int i;
   printf("Please enter a value:");
   scanf("%d", &i);
   printf( "\nYou entered: %d", i);
}
```

Field width e.g.:

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int d;
   printf("Enter max 5 numbers:");
   scanf("%5d",&d);
   printf("Entered number is %d",d);
```

```
    getch();
}
```

Input string:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char str[20];
    printf("Enter your name:");
    scanf("%s",&str);
    printf("Your name is %s",str);
    getch();
}
```

Reading mixed data types:

- In a single scanf call more than one of data can be read.
- Care should be taken to ensure that the input data items match the control specification.

E.g.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   char name[20];
   int roll;
   float marks;
   printf("Enter your name, roll number and marks:");
   scanf("%s%d%f",&name,&roll,&marks);
   printf("Name=%s\nRoll no.=%d\nMarks=%f",name,roll,marks);
   getch();
}
```

Formatted Output

- Refers to the output of data that has been arranged in a particular format.
- **printf()** is a built in function which is used to output data from the computer onto a standard device i.e. screen.
- General form:

  printf("control string",arg1,arg2,….argn);
- The control string cosists of four types of items:
  - Character that will be printed on the screen as they appear
  - Format specifications that define the output format for display of each item
  - Escape sequence character such as \n,\t etc.
  - Any combination of characters, format specifications and escape sequences.

Unformatted Functions

- Unformatted functions do not allow user to read or display data in desire format.
- These library functions basically deals with a single character or a string of character.
- The functions getchar(), putchar(), gets(), puts(), getch(), getche(), putche() are considered as unformatted functions.

- ***getchar()***

- Reads a character from a standard input device.
- It takes the form: **character_variable = getchar();**
- This function reads only single character at a time.

- ***putchar()***

- Displays a character to the standard output device.
- Its form: **putchar(character_variable)**
- This function displays only single character at a time.

E.g.

```
#include <stdio.h>
void main( )
{
   int c;
   printf("Enter a character:");
   /* Take a character as input and store it in variable c */
   c = getchar();
   /* display the character stored in variable c */
   putchar(c);
}
```

- ***gets()***

- used to read string of text, containing whitespace, until a new line character is encountered.
- General form: **gets(string_variable);**

- ***puts()***

- Used to display the string onto the terminal
- General form: **puts(string_variable);**

E.g.

```
#include<stdio.h>
void main()
{
   /* character array of length 100 */
   char str[100];
   printf("Enter a string:");
   gets( str );
   printf("The string you entered:");
   puts( str );
```

```
    getch();
}
```