# • Chapter 10 : Input / Output

## 10.1 Overview

Q : 10-01-01 : Describe Standard Input / Output Library (stdio.h) in C Language ?

**Answer :**

**Standard Input / Output Library** : In C, the standard input / output library provides functions to perform input and output operations. By standard input and output, we mean the keyboard and monitor respectively. In C, these input / output operations are performed by two standard input / output functions, these are printf( ) and scanf( ). These functions can be accessed by including the standard input / output library (stdio.h) in the program.

Q : 10-01-02 : Describe Standard Input / Output Library (stdio.h) Function printf ?



**Answer :**

**Standard Input / Output Library Function printf** : To see results of program execution, we must have a way to specify what variable values should be displayed. The standard I/O library (stdio.h) function printf (pronounced as print-eff) is used for formatted output. It takes as arguments a format string and an optional list of variables to output. The values of variables are displayed according to the specifications in the format string. The format string is a character string - nothing more and the variables are optional.

The printf( ) function will take the form :

printf(format string, var1, var2, var3, … ) ;
printf(format string) ;

**The Signatures of a Function** : Describes the number, order and type of its arguments, and the return type of the function (signature is identity).

**Example** : Write a program to calculate and print the area of a Rectangle.

```
#include<stdio.h>
void main (void)
{
        int         height, width, area ;
        height      = 5 ;
        width       = 4 ;
        area        = height * width ;
        printf ("Area of Rectangle = %d", area) ;
}
```

Program Outputs : Area of Rectangle = 20

In the above program, the first line is the variable declaration statement. In second and third lines, values are assigned to the variables height and width. Fourth line of code describes the arithmetic expression for calculating the area of the Rectangle and the result is assigned to the variable area. Fifth and the last line of code is the printf( ) statement. This displays result on the screen. In case of printf, the first parameter is

always a string (e.g.,"Area of Rectangle"), which should be enclosed in double quotes. This string is called the format string. Format string may include any number of format specifiers such as %d. The list of variables separated by commas, whose values are to be displayed in the result, will follow the format string.

Q : 10-01-03 : Describe Format Specifiers in Standard Input / Output Library (stdio.h) Function printf ?

**Answer :**

**Format Specifier** : Format specifiers specify the format in which the value of a variable should be displayed on the screen. Format specifiers are specified in the format string. For instance, in the above program the printf( ) statement contains the symbol %d, which is format specifier for the variable area. For different types of variables, different format specifiers are used :

| Symbol | Data Type |
|---|---|
| %d | int, short |
| %f | float |
| %lf | double |
| %e | float, double (Exponential Notation) |
| %g | Floating point (%f or %e; whichever is shorter) |
| %c | char |
| %s | Character string |
| %u | unsigned short, unsigned int |
| %x | Unsigned hexadecimal integers |
| %o | Unsigned octal integer |
| %i | Integers |
| %ld | long integer |

**Example** : Write a program that adds two floating point numbers and shows their sum on the screen.

```
#include<stdio.h>
void main (void)
{
    float    var1, var2, var3 ;
    var1     = 24.27 ;
    var2     = 41.50 ;
    var3     = var1 + var2 ;
    printf ("%f + %f = %f", var1, var2, var3) ;
}
```

Program Outputs : 24.27 + 41.5 = 65.77

Q : 10-01-04 : Describe Field-Width Specifiers in Standard Input / Output Library (stdio.h) Function printf ?

**Answer :**

**Field-Width Specifier** : In a C program, the number of columns used to display a value on the
screen is referred to as field-width. Field-width specifiers describe the number of columns that should be used to print a value.

**Formatting Integers** : Add a number between the % and d of the %d format specifier

in the printf format string, This number specifies the field-width or the number of columns to be used for the display of the value. The statement : printf ("Area = %4d", area) ; indicates that four columns will be used to display the value of area. Suppose the value of the variable area is 25. Two extra spaces will be padded before 25 (least significant) on the screen to complete the length of 4. The output of the above statement will be : Area = □□25. □ represents a blank space. This space will not be displayed as a printed character in actual output. In this way the value of 25, which requires two spaces to be displayed, will occupy four spaces (columns) on the screen. The reason is that the format specifier for area (%4d) allows spaces for four digits to be printed. Because the value of area is 25, therefore its two digits are right justified, preceded by two blank spaces.

| Value | Format Specifier | Displayed Output | Value | Format Specifier | Displayed Output |
|---|---|---|---|---|---|
| 786 | %4d | □786 | -786 | %4d | -786 |
| 786 | %5d | □□786 | -786 | %5d | □-786 |
| 786 | %6d | □□□786 | -786 | %6d | □□-786 |
| 786 | %1d | 786 | -786 | %2d | -786 |

The last row of this table shows that C expands the field width if it is too small for the integer value displayed.

**Formatting Floating Point Numbers** : We need to indicate both the total field width needed and the number of decimal places desired. The total field width should be large enough to accommodate all digits before and after the decimal point e.g., to display 15.245 and 0.12 the total field width should be six and four respectively.

**Important Note** : For numbers smaller than zero, a zero is always printed before the decimal point. Therefore the total field width should include a space for the decimal point as well as for the minus sign if the number can be negative. The general form for the format specifier for a floating point value will be %m.nf, where m represents the total field width, and n represents the desired number of decimal places. The statement : printf ("Height = %6.2f", height) ; indicates that the total field width for the value of the variable height is 6, and the accuracy is of two decimal places. The value of height will be rounded off to two decimal places and will be displayed right justified in 6 columns. While being rounded off, if the third digit of the value's fractional part is 5 or greater, the second digit is increased by one otherwise the third digit is discarded.

| Value | Format Specifier | Displayed Output | Value | Format Specifier | Displayed Output |
|---|---|---|---|---|---|
| -25.41 | %6.2f | -25.41 | .123 | %6.2f | 0.12 |
| 3.14159 | %5.2f | 3.14 | 3.14159 | %4.2f | 3.14 |
| 3.14159 | %3.2f | 3.14 | 3.14159 | %5.1f | 3.1 |
| 3.14159 | %5.3f | 3.142 | 3.14159 | %8.5f | 3.14159 |
| .6789 | %4.2f | 0.69 | -0.007 | %4.2f | -0.01 |
| -.007 | %8.3f | -0.007 | -0.007 | %8.5f | -0.00700 |
| -.007 | %.3f | -0.007 | -3.14159 | %.4f | -3.1416 |

Q : 10-01-05 : Explain Escape Sequences in Standard Input / Output Library (stdio.h) Function printf( ) ?

**Answer :**

**Escape Sequences** : Escape sequences are characters which are specified in the format string of the printf statement in combination with a backslash (\). These cause an escape from the normal interpretation of a string so that the next character is recognized as having a special meanings.

**Example** : Write a program that will demonstrate the use of escape sequences.

```
#include<stdio.h>
void main (void)
{
    printf ("Name\t\tRoll No\t\tMarks") ;
    printf ("\n---------------------------------------") ;
    printf ("\nAmir\t\t    78\t\t425") ;
    printf ("\nTahir\t\t    23\t\t385") ;
}
```

Program Outputs :

```
Name            Roll No            Marks
---------------------------------------
Amir              78                425
Tahir             23                385
```

We used two escape sequences. These are \t and \n. The escape sequence \n causes the text to print from the start of the next line, whereas \t inserts a tab space between two words. In addition to new line and tab escape sequences; there are some others as well :

| Escape Sequence | Purpose |
|---|---|
| \n | New Line |
| \t | Tab |
| \b | Backspace |
| \r | Carriage Return (Enter Key) |
| \f | Form feed |
| \' | Single Quote |
| \" | Double Quote |
| \\ | Backslash |
| \xdd | ASCII code in hexadecimal notation (each d represents a digit) |
| \ddd | ASCII code in octal notation (each d represents a digit) |

The escape sequence \b causes the cursor to move one space left, the form-feed (\f) moves to the next page on printer.

**Important Note** : It is important to note that one can not display a single or double quote on the screen without using the escape sequences \‘ and \“. The reason is that the format string of the printf function is enclosed in a double quote. When a double quote is specified in the format string, it is treated as the closing double quote. That’s why, single and double quotes are always written with backslash. Statement : printf (“Escape Sequence is a \“Cool\” feature of C.”) ;
The output is : Escape Sequence is a “Cool” feature of C.

## 10.2 Scanf Function

Q : 10-02-01 : Explain Standard Input / Output Library (stdio.h) Function scanf ?

**Answer :**

**Standard Input / Output Library (stdio.h) Function scanf** : Most of the programs are interactive in nature. C is featured with a range of functions to accept user input in variety of forms. The scanf (pronounced as scan-eff) function is versatile as it is equally good for numeric as well as string input.
It takes as arguments a format string and a list of variables to hold the input values.
Here is the syntax of function scanf :
Example : Write a program to get the distance in kilometers from user, convert it into meters and display on the screen.

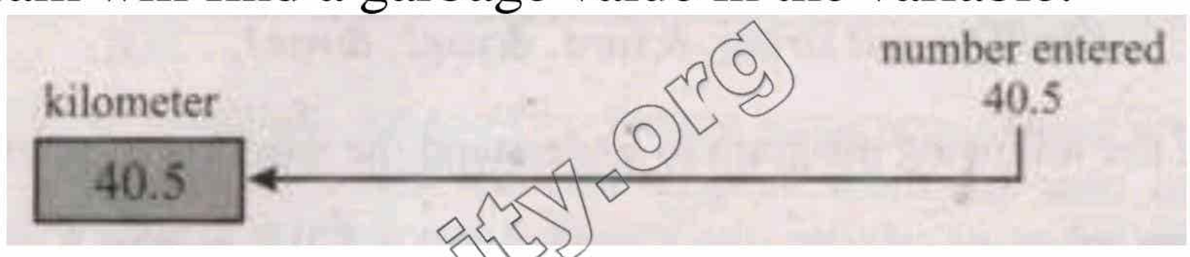
```
#include<stdio.h>
void main (void)
{
        double meter, kilometer ;
        // Ask the user to enter kilometers
        printf (“\nEnter distance in kilometers => ”) ;
        // Take input
        scanf (“%1f”, &kilometer) ;
        meter = kilometer * 1000 ;
```

```
                    printf ("\n%1f kilometers = %1f meters", kilometer,
                meter) ;
                }
```

Program Outputs : Enter distance in kilometers => 90.13
90.130000 kilometers = 90130.000000 meters

First line is the declaration of variables meter and kilometer. The next executable statement is printf, which displays a message for the user to enter distance in kilometers. The next is the scanf statement. When the program reaches this line of code, the flow of execution stops until the user enters a value. The format string of scanf is "%lf" which tells the scanf what kind of data to copy into variable kilometer. The format string of scanf consists of a list of format specifiers only; no other value or text can be specified in it.

**Important Note** : Instead of the variable name, the scanf function requires address of the variable to store the input value into it. In the call to scanf the name of variable kilometer is preceded by an ampersand character (&). In C, & is actually the address of operator. In scanf, the address of operator (&) tells the scanf function the address of the variable where the input value is to be stored. If & is omitted, the scanf will not be able to locate the variable in memory, hence it will be unable to store the value into the variable and the program will find a garbage value in the variable.




## 10.3 Character Input

Q : 10-03-01 : Explain Standard Library Functions for Character Input ?

**Answer :**

**Standard Library Functions for Character Input** : In C, there are many functions to accept character input. The versatile scanf can also be used for this purpose. But scanf requires pressing the return key at the end of input value. In some cases, it. is desirable to input characters without pressing the return key. For example, in a game while controlling the movement of a space ship through arrow keys we can't afford to press return key each time after pressing an arrow key. To overcome such situations, C is equipped with many other functions specialized for character input. Function getch and getche are good examples. These are part of the conio.h (Console Input Output) library.

**getch and getche Functions :** The getch and getche functions are very handy in character manipulation. In contrast to the getch function which does not echo the character typed, the getche function echo the typed character. Both of these functions do not accept any argument. However they return the typed character to the calling function. One does not need to press the return key (ENTER key) after typing the character. The moment a character is typed, it is immediately returned by the function to the calling module.

**Important Note** : Character Constants are always specified within single quotations e.g, 'a', 'x' etc.

**Example** : Write a program that displays the ASCII code of the character typed by the user.

```
#include<stdio.h>
#include<conio.h>
void main (void)
{
        char   ch ;
        // Ask the user to enter any key
        printf ("\nPlease Type a Character => ") ;
        // Take input
        ch      = getche() ;
        printf ("\nThe ASCII code for \'%c\' is %d", ch, ch) ;
}
```

Program Outputs : Please Type a Character => g
The ASCII code for 'g' is 103

**Important Note** : In this program we include a new header file conio.h. This file contains
the definition of functions getch() and getche. In this program, the statement ch = getche( ) ; can be replaced with the statement ch = getch( ) ; In the later case, the typed character will not be shown on the screen and the output will be :

**Program Outputs** : Please Type a Character =>
The ASCII code for 'g' is 103

When the 3rd line of the program is executed, it waits for a character to be typed. As soon as a character is typed, the very next line executes immediately without waiting for the return key to be typed. And the function getche() returns the typed character to the main function, where it is assigned to the variable ch.

## Exercise 10

**Q-9**. Write a program that asks the user to enter the radius of a circle and then computes and displays the circle's area. Use the formula : ($\pi$ $r^2$) : area = PI x radius x radius.

where PI or $\pi$ is the constant value of 3.14159. (Note: Define a constant macro PI with #define directive).

**Answer :**

```
#include<stdio.h>
#define PI 3.14159
void main (void)
{
        float radius ;
        double area ;
        // Ask the user to enter radius
        printf ("\nEnter Radius of Circle => ") ;
        // Take input
        scanf ("%f", &radius) ;
```

```
                area = PI * radius * radius ;
                printf ("\nArea of Circle = %1f", area) ;
        }
```

**Q-10**. Write a program that stores the values 'A', 'U', 3.456E10 and 50 in separate memory cells. Your program should get the first three values as input data, but use an assignment statement to store the last value.

**Answer :**

```
#include<stdio.h>
void main (void)
{
        char ch1, ch2 ;
        float real_num ;
        int i ;
        // Ask the user to enter ch1 and ch2
        printf ("\nEnter First Character => ") ;
        scanf ("%c", &ch1) ;
        printf ("\nEnter Second Character => ") ;
        scanf ("%c", &ch2) ;
        printf ("\nEnter Real Number in Exponential Format
        => ") ;
        scanf ("%e", &real_num) ;
        i = 50 ;
        printf ("\nValue of Intger = %d", i) ;
}
```

**Q-11**. Write a program that converts a temperature in degrees Fahrenheit to degree1 Celsius. For conversion, use the formula : Celsius = 5/9 (Fahrenheit – 32).

**Answer :**

```
#include<stdio.h>
void main (void)
{
        float Celsius, Fahrenheit ;
        // Ask the user to enter Fahrenheit
        printf ("\nEnter Temperature in Fahrenheit => ") ;
        // Take input
        scanf ("%f", &Fahrenheit) ;
        Celsius = (5 / 9) * (Fahrenheit – 32) ;
        printf ("\n%.2f Fahrenheit = %.2f Celsius",
        Fahrenheit, Celsius) ;
}
```

**Q-12**. Write a program that takes a positive number with a fractional part and round it to two decimal places. For example, 25.4851 would round to 25.49, 62.4431 would round to 32.44.

**Answer :**

```
#include<stdio.h>
#include<math.h>
void main (void)
{
        float positive_real_num ;
        // Ask the user to enter a positive real number
        printf (“\nEnter a Positive Real Number => ”) ;
        // Take input
        scanf (“%f”, &positive_real_num) ;

        printf (“\n%.2f is a Positive Real Number”,
        positive_real_num) ;
}
```