**Quick History of C**

• Developed at Bell Laboratories in the early seventies by Dennis Ritchie.

• Born out of two other languages – BCPL(Basic Control Programming Language) and B.

• C introduced such things as character types, floating point arithmetic, structures, unions and the preprocessor.

• The principal objective was to devise a language that was easy enough to understand to be "high-level" – i.e. understood by general programmers, but low-level enough to be applicable to the writing of systems-level software.

• The language should abstract the details of how the computer achieves its tasks in such a way as to ensure that C could be portable across different types of computers, thus allowing the UNIX operating system to be compiled on other computers with a minimum of re-writing.

• C as a language was in use by 1973, although extra functionality, such as new types, was introduced up until 1980.

• In 1978, Brian Kernighan and Dennis M. Ritchie wrote the seminal work The C Programming Language, which is now the standard reference book for C.

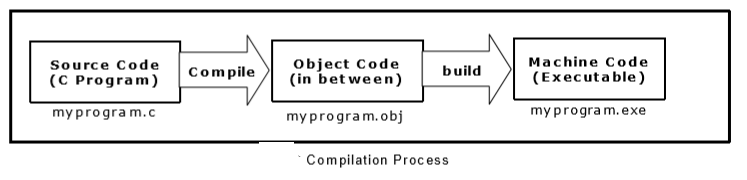
• A formal ANSI standard for C was produced in 1989.

• In 1986, a descendant of C, called C++ was developed by Bjarne Stroustrup, which is in wide use today. Many modern languages such as C#, Java and Perl are based on C and C++.

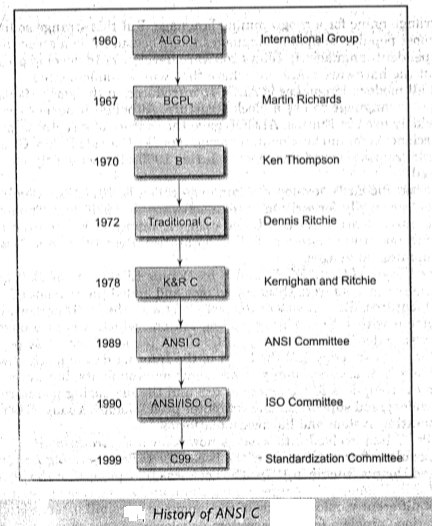
• Using C language scientific, business and system-level applications can be developed easily.

**The Compile Process**

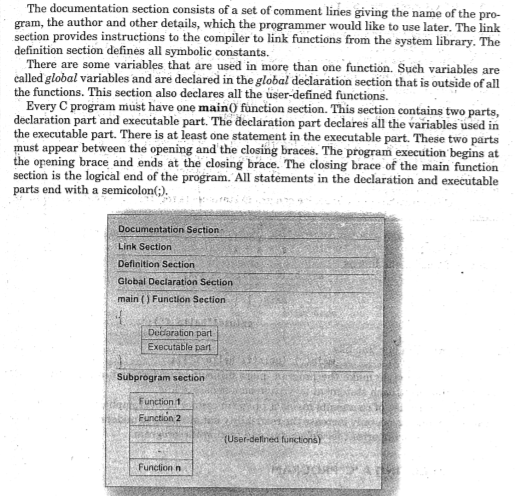
You type in your program as C source code, compile it (or try to run it from your programming editor, which will typically compile and build automatically) to turn it into object code, and then build the program using a linker program to create an executable program that contains instructions written in machine code, which can be understood and run on the computer that you are working on.

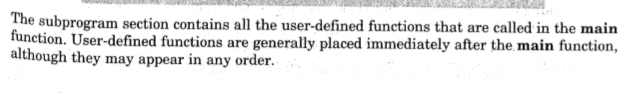


**History of ANSI C**

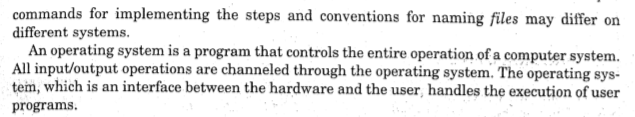
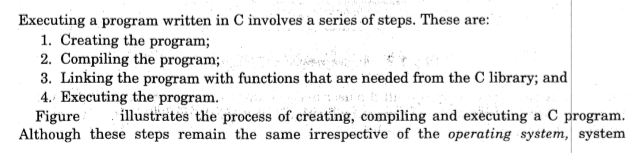
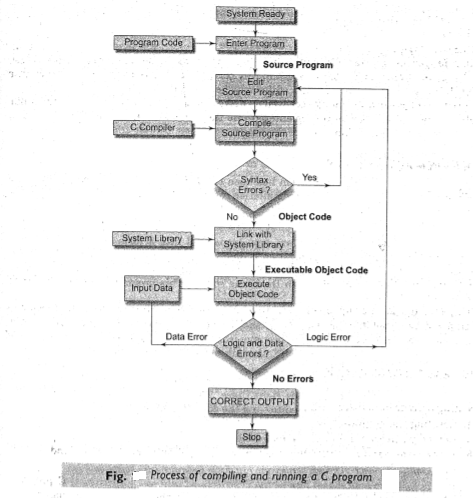


**Basic Structure of C programs**

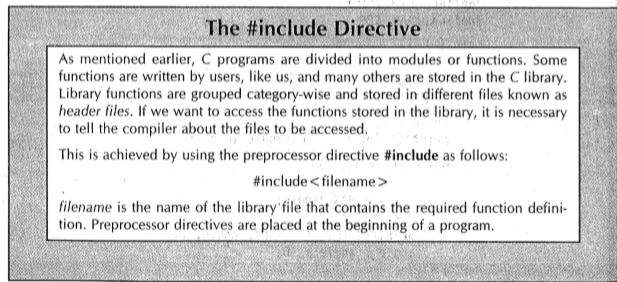
****

****

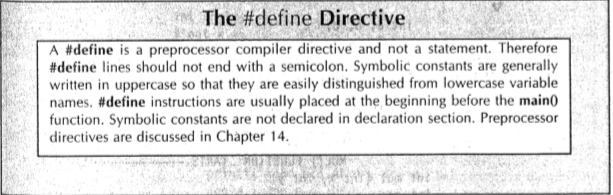
**Process of C program Execution**



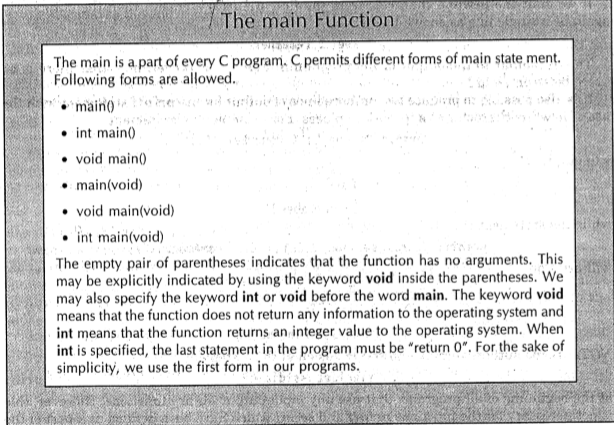
**#include directive**



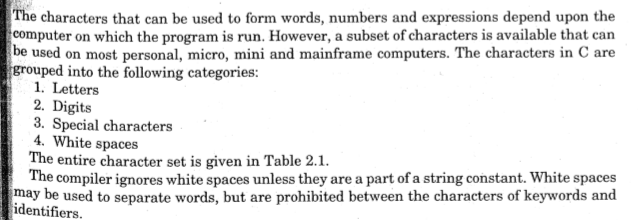
**#define directive**

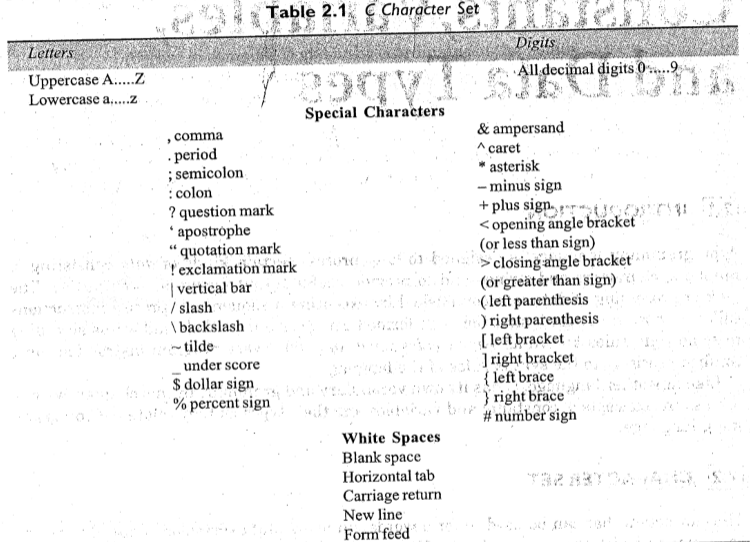


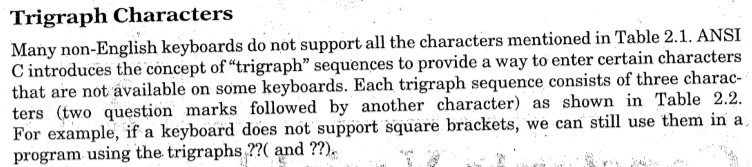
**Main() function**

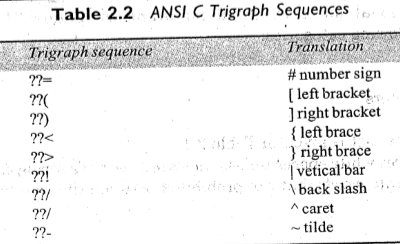
****

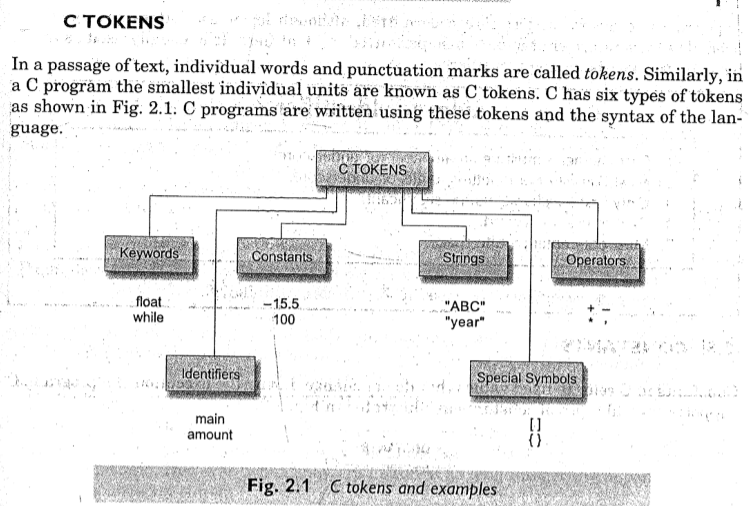
**Character set in C**

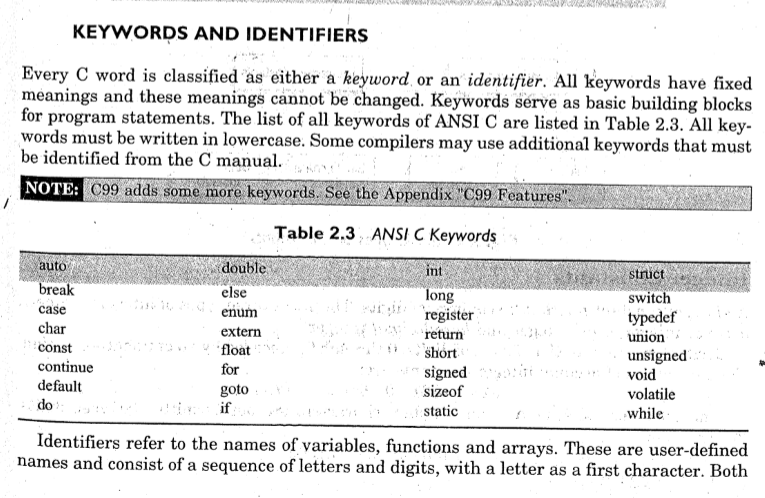
****

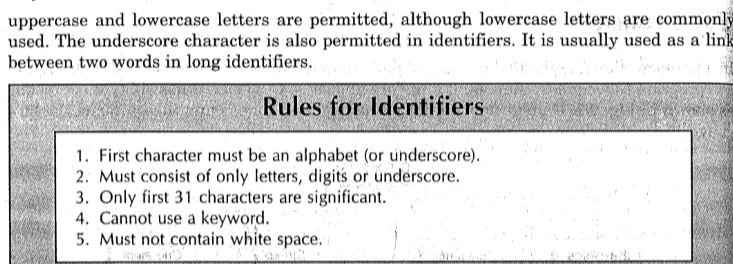
****

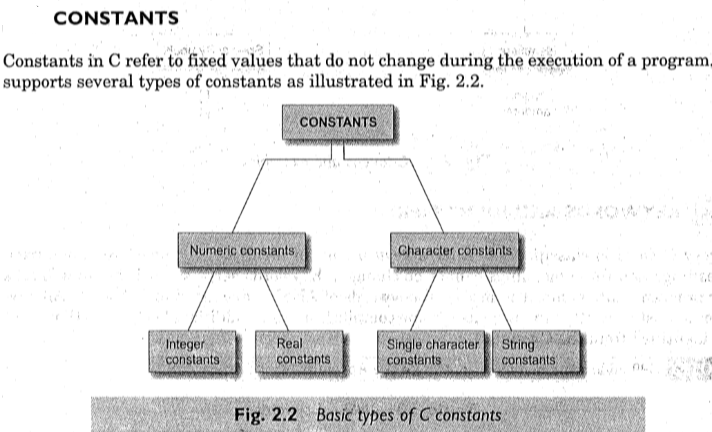
****

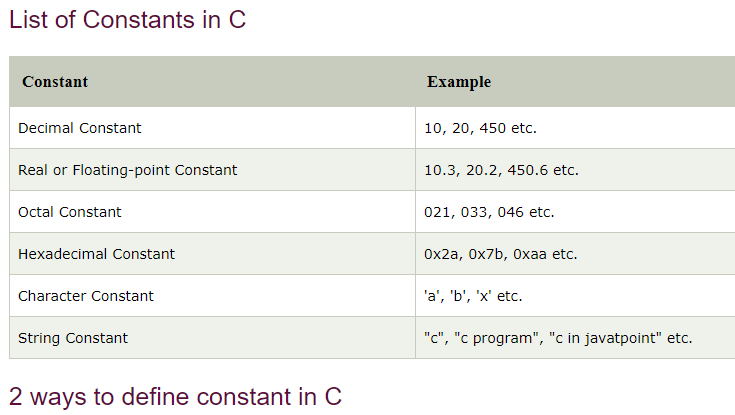
****

****

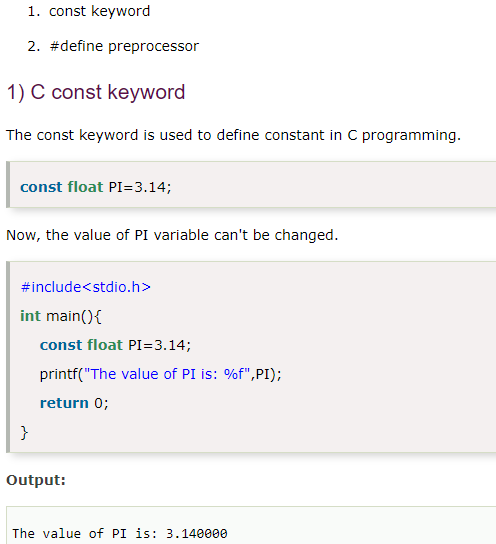
****

****

****

****

There are two ways to define constant in [C programming](https://www.javatpoint.com/c-programming-language-tutorial).

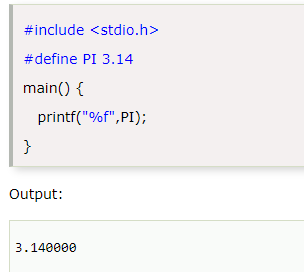
****

# 2) C #define

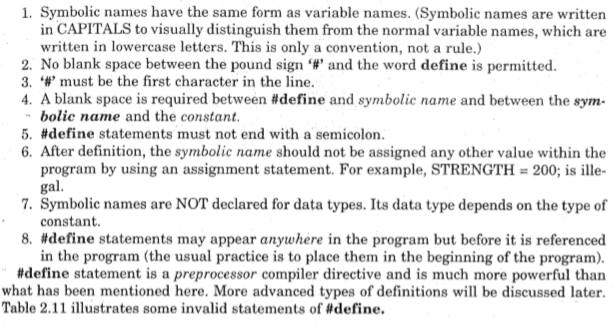
The #define preprocessor directive is used to define constant or micro substitution. It can use any basic data type.

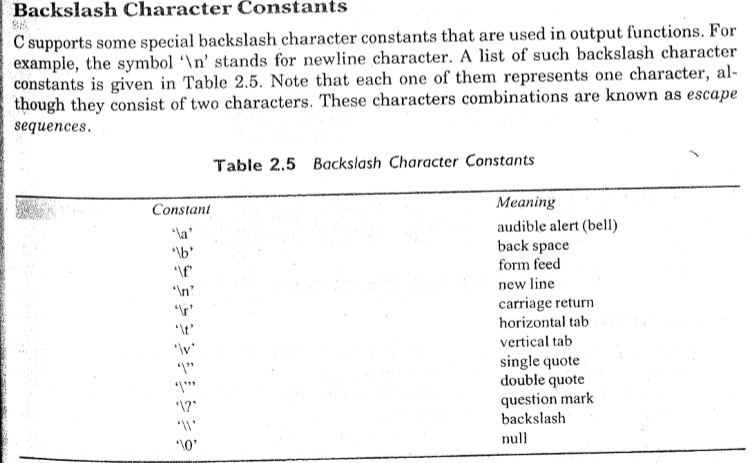
Syntax: #define token value

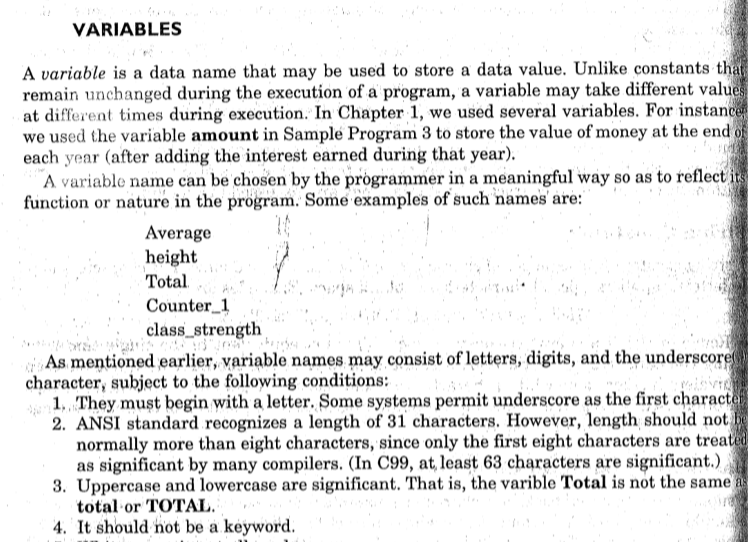
Let's see an example of #define to define a constant.

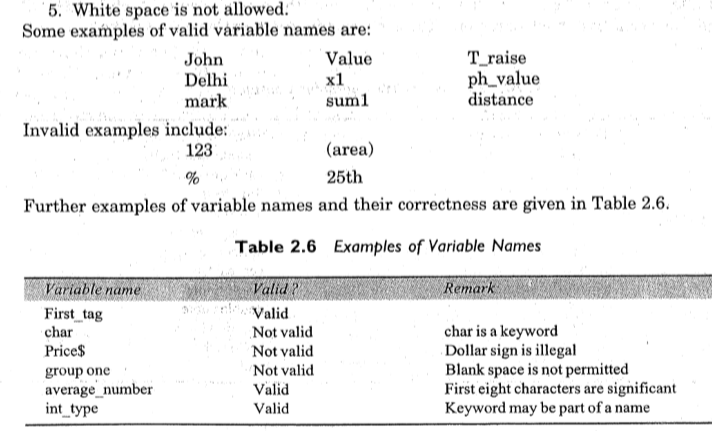


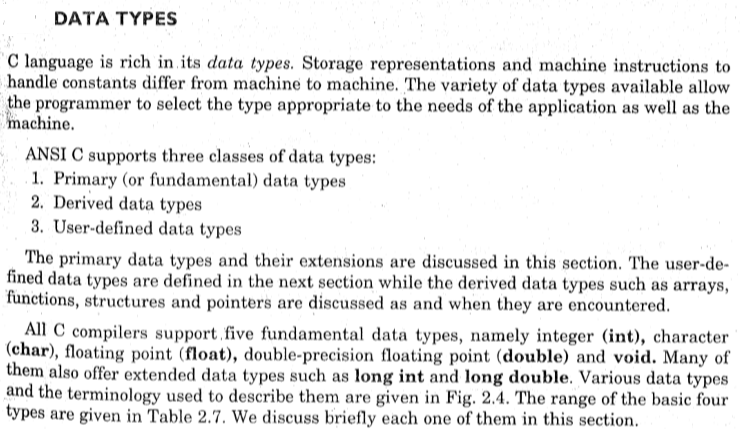
Rules to define symbolic constant using #define

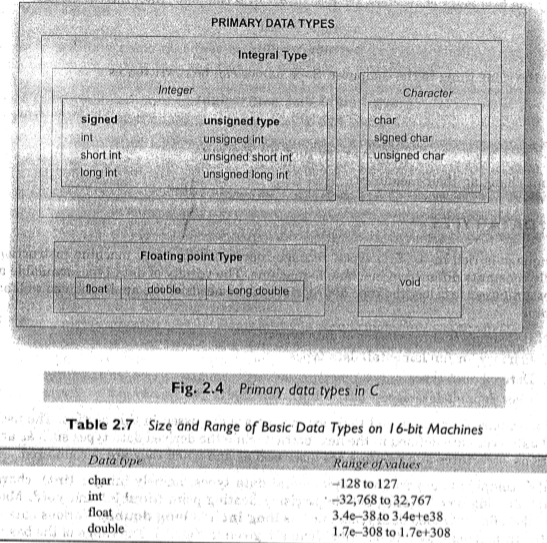


****

****

****

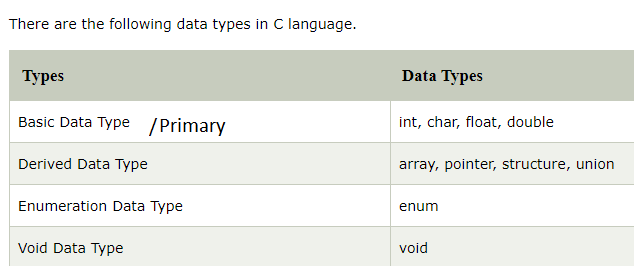
****

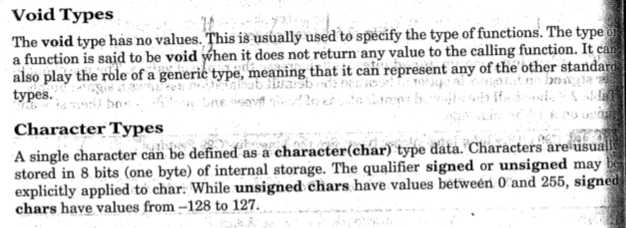
****

The memory size of the basic data types may change according to 32 or 64-bit operating system.

Let's see the basic data types. Its size is given **according to 32-bit architecture**.

|  |  |  |
| --- | --- | --- |
| Data Types | Memory Size | Range |
| char | 1 byte | −128 to 127 |
| signed char | 1 byte | −128 to 127 |
| unsigned char | 1 byte | 0 to 255 |
| short | 2 byte | −32,768 to 32,767 |
| signed short | 2 byte | −32,768 to 32,767 |
| unsigned short | 2 byte | 0 to 65,535 |
| int | 2 byte | −32,768 to 32,767 |
| signed int | 2 byte | −32,768 to 32,767 |
| unsigned int | 2 byte | 0 to 65,535 |
| short int | 2 byte | −32,768 to 32,767 |
| signed short int | 2 byte | −32,768 to 32,767 |
| unsigned short int | 2 byte | 0 to 65,535 |
| long int | 4 byte | -2,147,483,648 to 2,147,483,647 |
| signed long int | 4 byte | -2,147,483,648 to 2,147,483,647 |
| unsigned long int | 4 byte | 0 to 4,294,967,295 |
| float | 4 byte |  |
| double | 8 byte |  |
| long double | 10 byte |  |

****

****

# C Format Specifier

The Format specifier is a string used in the formatted input and output functions. The format string determines the format of the input and output. The format string always starts with a '%' character.

**The commonly used format specifiers in printf() function are:**

|  |  |
| --- | --- |
| Format specifier | Description |
| %d or %i | It is used to print the signed integer value where signed integer means that the variable can hold both positive and negative values. |
| %u | It is used to print the unsigned integer value where the unsigned integer means that the variable can hold only positive value. |
| %o | It is used to print the octal unsigned integer where octal integer value always starts with a 0 value. |
| %x | It is used to print the hexadecimal unsigned integer where the hexadecimal integer value always starts with a 0x value. In this, alphabetical characters are printed in small letters such as a, b, c, etc. |
| %X | It is used to print the hexadecimal unsigned integer, but %X prints the alphabetical characters in uppercase such as A, B, C, etc. |
| %f | It is used for printing the decimal floating-point values. By default, it prints the 6 values after '.'. |
| %e/%E | It is used for scientific notation. It is also known as Mantissa or Exponent. |
| %g | It is used to print the decimal floating-point values, and it uses the fixed precision, i.e., the value after the decimal in input would be exactly the same as the value in the output. |
| %p | It is used to print the address in a hexadecimal form. |
| %c | It is used to print the unsigned character. |
| %s | It is used to print the strings. |
| %ld | It is used to print the long-signed integer value. |
| Format specifier | Description |
| %d or %i | It is used to print the signed integer value where signed integer means that the variable can hold both positive and negative values. |
| %u | It is used to print the unsigned integer value where the unsigned integer means that the variable can hold only positive value. |
| %o | It is used to print the octal unsigned integer where octal integer value always starts with a 0 value. |
| %x | It is used to print the hexadecimal unsigned integer where the hexadecimal integer value always starts with a 0x value. In this, alphabetical characters are printed in small letters such as a, b, c, etc. |
| %X | It is used to print the hexadecimal unsigned integer, but %X prints the alphabetical characters in uppercase such as A, B, C, etc. |
| %f | It is used for printing the decimal floating-point values. By default, it prints the 6 values after '.'. |
| %e/%E | It is used for scientific notation. It is also known as Mantissa or Exponent. |
| %g | It is used to print the decimal floating-point values, and it uses the fixed precision, i.e., the value after the decimal in input would be exactly the same as the value in the output. |

**Enumerated (Enum) data type in C**

The enum in C is also known as the enumerated type. It is a user-defined data type that consists of integer values, and it provides meaningful names to these values. The use of enum in C makes the program easy to understand and maintain. The enum is defined by using the enum keyword.

The following is the way to define the enum in C:

**enum** flag{integer\_const1, integer\_const2,.....integter\_constN};

In the above declaration, we define the enum named as flag containing 'N' integer constants. The default value of integer\_const1 is 0, integer\_const2 is 1, and so on. We can also change the default value of the integer constants at the time of the declaration.

**For example:** **enum** fruits{mango, apple, strawberry, papaya};

The default value of mango is 0, apple is 1, strawberry is 2, and papaya is 3. If we want to change these default values, then we can do as given below:

**enum** fruits{

mango=2,

apple=1,

strawberry=5,

papaya=7,

};

**Enumerated type declaration**

As we know that in C language, we need to declare the variable of a pre-defined type such as int, float, char, etc. Similarly, we can declare the variable of a user-defined data type, such as enum. Let's see how we can declare the variable of an enum type.

**Suppose we create the enum of type status as shown below:**

enum status{false,true};

**Now, we create the variable of status type:**

enum status s; // creating a variable of the status type.

In the above statement, we have declared the 's' variable of type status.

**To create a variable, the above two statements can be written as:**

enum status{false,true} s;

In this case, the default value of false will be equal to 0, and the value of true will be equal to 1.

**Let's create a simple program of enum.**

#include <stdio.h>

enum weekdays{Sunday=1, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

int main()

{

enum weekdays w; // variable declaration of weekdays type

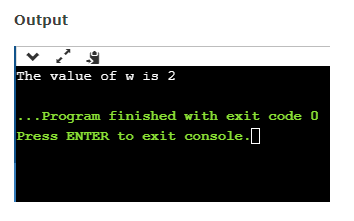
w=Monday; // assigning value of Monday to w.

printf("The value of w is %d",w);

return 0;

}

In the above code, we create an enum type named as weekdays, and it contains the name of all the seven days. We have assigned 1 value to the Sunday, and all other names will be given a value as the previous value plus one.

****

# Formatted input - scanf() and Formatted output - printf() in c

The printf() and scanf() functions are used for input and output in C language. Both functions are inbuilt library functions, defined in stdio.h (header file).

### **printf() function**

The **printf() function** is used for output. It prints the given statement to the console. Statement to be printed is either message of variable value.

**The syntax of printf() function is given below:**

**printf("format string",argument\_list);**

The **format string/ control string** can be %d (integer), %c (character), %s (string), %f (float) etc. **argument\_list** is

### **scanf() function**

### 

### **Program to print cube of given number**

Let's see a simple example of c language that gets input from the user and prints the cube of the given number.

#include<stdio.h>

**Void** main()

{

**int** number;

printf("enter a number:");

scanf("%d",&number);

printf("cube of number is:%d ",number\*number\*number);

}

**Output**

enter a number:5

cube of number is:125

The **scanf("%d",&number)** statement reads integer number from the console and stores the given value in number variable.

The **printf("cube of number is:%d ",number\*number\*number)** statement prints the cube of number on the console.

# Comments in C

Comments in C language are used to provide information about lines of code. It is widely used for documenting code. There are 2 types of comments in the C language.

1. Single Line Comments
2. Multi-Line Comments

## **Single Line Comments**

Single line comments are represented by double slash \\. Let's see an example of a single line comment in C.

#include<stdio.h>

**void** main()

{

    //printing information

    printf("Hello C");

}

Output:

Hello C

Even you can place the comment after the statement. For example:

printf("Hello C");//printing information

## **Mult Line Comments**

Multi-Line comments are represented by slash asterisk \\* ... \*\. It can occupy many lines of code, but it can't be nested. Syntax:

/\*

code

to be commented

\*/

Let's see an example of a multi-Line comment in C.

#include<stdio.h>

**void** main()

{

    /\*printing information

      Multi-Line Comment\*/

    printf("Hello C");

}

Output:

Hello C

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***