**C Programming Language**

The C Language is developed by Dennis Ritchie for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.

C programming is considered as the base for other programming languages, that is why it is known as mother language.

It can be defined by the following ways:

1. Mother language
2. System programming language
3. Procedure-oriented programming language
4. Structured programming language
5. Mid-level programming language

**1) C as a mother language**

C language is considered as the mother language of all the modern programming languages because **most of the compilers, JVMs, Kernels, etc. are written in C language**, and most of the programming languages follow C syntax, for example, C++, Java, C#, etc.

It provides the core concepts like the [array](https://www.javatpoint.com/c-array)

, [strings](https://www.javatpoint.com/c-strings)

, [functions](https://www.javatpoint.com/functions-in-c)

, [file handling](https://www.javatpoint.com/file-handling-in-c)

, etc. that are being used in many languages like [C++](https://www.javatpoint.com/cpp-tutorial)

, [Java](https://www.javatpoint.com/java-tutorial)

, [C#](https://www.javatpoint.com/c-sharp-tutorial)

, etc.

**2) C as a system programming language**

A system programming language is used to create system software. C language is a system programming language because it **can be used to do low-level programming (for example driver and kernel)**. It is generally used to create hardware devices, OS, drivers, kernels, etc. For example, Linux kernel is written in C.

It can't be used for internet programming like Java, .Net, PHP, etc.

**3) C as a procedural language**

A procedure is known as a function, method, routine, subroutine, etc. A procedural language **specifies a series of steps for the program to solve the problem**.

A procedural language breaks the program into functions, data structures, etc.

C is a procedural language. In C, variables and function prototypes must be declared before being used.

**4) C as a structured programming language**

A structured programming language is a subset of the procedural language. **Structure means to break a program into parts or blocks** so that it may be easy to understand.

In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

**5) C as a mid-level programming language**

C is considered as a middle-level language because it **supports the feature of both low-level and high-level languages**. C language program is converted into assembly code, it supports pointer arithmetic (low-level), but it is machine independent (a feature of high-level).

A **Low-level language** is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand.

A **High-Level language** is not specific to one machine, i.e., machine independent. It is easy to understand.

# History of C Language

**History of C language** is interesting to know. Here we are going to discuss a brief history of the c language.

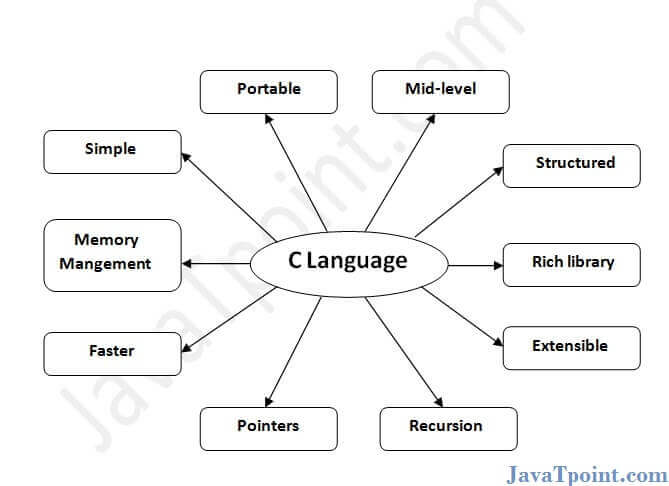
**C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

**Dennis Ritchie** is known as the **founder of the c language**.

It was developed to overcome the problems of previous languages such as B, BCPL, etc.HTML Tutorial

Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL.

**Features of C Language**



C is the widely used language. It provides many **features** that are given below.

1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed
8. Pointers
9. Recursion
10. Extensible

**1) Simple**

C is a simple language in the sense that it provides a **structured approach** (to break the problem into parts), **the rich set of library functions**, **data types**, etc.

**2) Machine Independent or Portable**

Unlike assembly language, c programs **can be executed on different machines** with some machine specific changes. Therefore, C is a machine independent language.

**3) Mid-level programming language**

Although, C is **intended to do low-level programming**. It is used to develop system applications such as kernel, driver, etc. It **also supports the features of a high-level language**. That is why it is known as mid-level language.

**4) Structured programming language**

C is a structured programming language in the sense that **we can break the program into parts using functions**. So, it is easy to understand and modify. Functions also provide code reusability.

**5) Rich Library**

C **provides a lot of inbuilt functions** that make the development fast.

**6) Memory Management**

It supports the feature of **dynamic memory allocation**. In C language, we can free the allocated memory at any time by calling the **free()** function.

**7) Speed**

The compilation and execution time of C language is fast since there are lesser inbuilt functions and hence the lesser overhead.

**8) Pointer**

C provides the feature of pointers. We can directly interact with the memory by using the pointers. We **can use pointers for memory, structures, functions, array**, etc.

**9) Recursion**

In C, we **can call the function within the function**. It provides code reusability for every function. Recursion enables us to use the approach of backtracking.

**10) Extensible**

C language is extensible because it **can easily adopt new features**.

**First C Program**

Before starting the abcd of C language, you need to learn how to write, compile and run the first c program.

To write the first c program, open the C console and write the following code:

1. #include <stdio.h>
2. **int** main(){
3. printf("Hello C Language");
4. **return** 0;
5. }

**#include <stdio.h>** includes the **standard input output** library functions. The printf() function is defined in stdio.h .

**int main()** The **main() function is the entry point of every program** in c language.

Prime Ministers of India | List of Prime Minister of India (1947-2020)

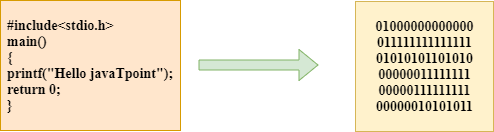
**printf()** The printf() function is **used to print data** on the console.

**return 0** The return 0 statement, returns execution status to the OS. The 0 value is used for successful execution and 1 for unsuccessful execution.

# Compilation process in c

## What is a compilation?

The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.



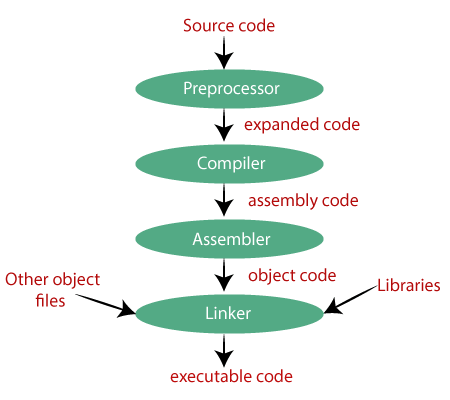
The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps, i.e., Pre-processing, Compiling, Assembling, and Linking.

The preprocessor takes the source code as an input, and it removes all the comments from the source code. The preprocessor takes the preprocessor directive and interprets it. For example, if **<stdio.h>,** the directive is available in the program, then the preprocessor interprets the directive and replace this directive with the content of the **'stdio.h'** file.

The following are the phases through which our program passes before being transformed into an executable form:

You will fix your code later. Just a temporarily fix, right? (consequences of technical debt)

* **Preprocessor**
* **Compiler**
* **Assembler**
* **Linker**



### **Preprocessor**

The source code is the code which is written in a text editor and the source code file is given an extension ".c". This source code is first passed to the preprocessor, and then the preprocessor expands this code. After expanding the code, the expanded code is passed to the compiler.

### **Compiler**

The code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code. Or we can say that the C compiler converts the pre-processed code into assembly code.

### **Assembler**

The assembly code is converted into object code by using an assembler. The name of the object file generated by the assembler is the same as the source file. The extension of the object file in DOS is '.obj,' and in UNIX, the extension is 'o'. If the name of the source file is **'hello.c',** then the name of the object file would be 'hello.obj'.

### **Linker**

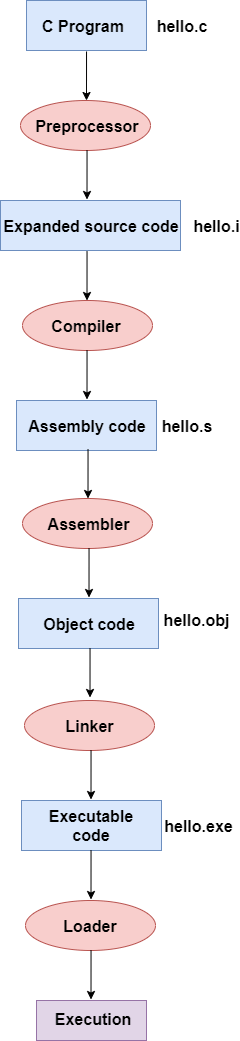
Mainly, all the programs written in C use library functions. These library functions are pre-compiled, and the object code of these library files is stored with '.lib' (or '.a') extension. The main working of the linker is to combine the object code of library files with the object code of our program. Sometimes the situation arises when our program refers to the functions defined in other files; then linker plays a very important role in this. It links the object code of these files to our program. Therefore, we conclude that the job of the linker is to link the object code of our program with the object code of the library files and other files. The output of the linker is the executable file. The name of the executable file is the same as the source file but differs only in their extensions. In DOS, the extension of the executable file is '.exe', and in UNIX, the executable file can be named as 'a.out'. For example, if we are using printf() function in a program, then the linker adds its associated code in an output file.

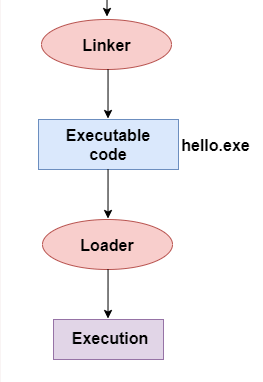
**Let's understand through an example.**

**hello.c**

1. #include <stdio.h>
2. **int** main()
3. {
4. printf("Hello javaTpoint");
5. **return** 0;
6. }

**Now, we will create a flow diagram of the above program:**



****

**In the above flow diagram, the following steps are taken to execute a program:**

* Firstly, the input file, i.e., **hello.c,** is passed to the preprocessor, and the preprocessor converts the source code into expanded source code. The extension of the expanded source code would be **hello.i.**
* The expanded source code is passed to the compiler, and the compiler converts this expanded source code into assembly code. The extension of the assembly code would be **hello.s.**
* This assembly code is then sent to the assembler, which converts the assembly code into object code.
* After the creation of an object code, the linker creates the executable file. The loader will then load the executable file for the execution.

# printf() and scanf() 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.

The syntax of printf() function is given below:

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

The **format string** can be %d (integer), %c (character), %s (string), %f (float) etc.

### **scanf() function**

The **scanf() function** is used for input. It reads the input data from the console.

scanf("format string",argument\_list);

### **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.

1. #include<stdio.h>
2. **int** main(){
3. **int** number;
4. printf("enter a number:");
5. scanf("%d",&number);
6. printf("cube of number is:%d ",number\*number\*number);
7. **return** 0;
8. }

**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.

# 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. |

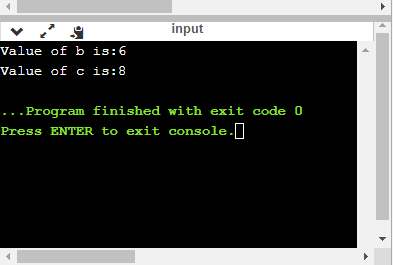
**Let's understand the format specifiers in detail through an example.**

* **%d**

1. **int** main()
2. {
3. **int** b=6;
4. **int** c=8;
5. printf("Value of b is:%d", b);
6. printf("\nValue of c is:%d",c);
8. **return** 0;
9. }

In the above code, we are printing the integer value of b and c by using the %d specifier.

**Output**

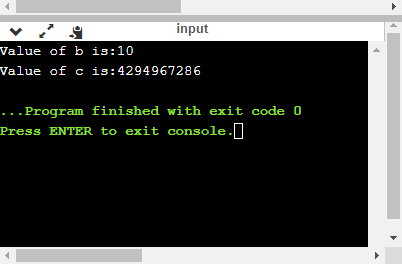


* **%u**

1. **int** main()
2. {
3. **int** b=10;
4. **int** c= -10;
5. printf("Value of b is:%u", b);
6. printf("\nValue of c is:%u",c);
8. **return** 0;
9. }

In the above program, we are displaying the value of b and c by using an unsigned format specifier, i.e., %u. The value of b is positive, so %u specifier prints the exact value of b, but it does not print the value of c as c contains the negative value.

**Output**

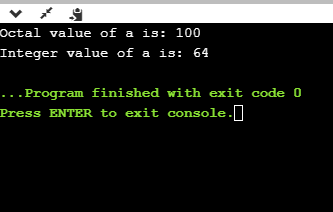


* **%o**

1. **int** main()
2. {
3. **int** a=0100;
4. printf("Octal value of a is: %o", a);
5. printf("\nInteger value of a is: %d",a);
6. **return** 0;
7. }

In the above code, we are displaying the octal value and integer value of a.

**Output**

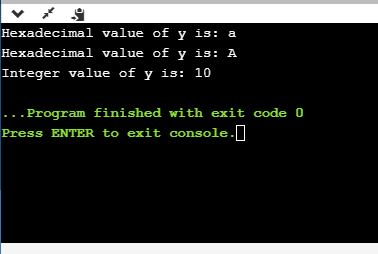


* **%x and %X**

1. **int** main()
2. {
3. **int** y=0xA;
4. printf("Hexadecimal value of y is: %x", y);
5. printf("\nHexadecimal value of y is: %X",y);
6. printf("\nInteger value of y is: %d",y);
7. **return** 0;
8. }

In the above code, y contains the hexadecimal value 'A'. We display the hexadecimal value of y in two formats. We use %x and %X to print the hexadecimal value where %x displays the value in small letters, i.e., 'a' and %X displays the value in a capital letter, i.e., 'A'.

**Output**

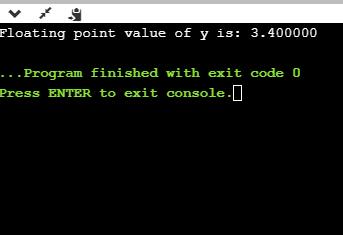


* **%f**

1. **int** main()
2. {
3. **float** y=3.4;
4. printf("Floating point value of y is: %f", y);
5. **return** 0;
6. }

The above code prints the floating value of y.

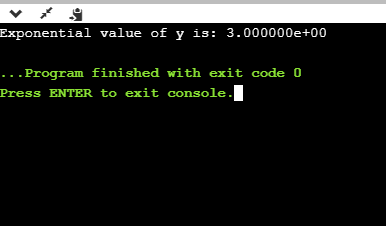
**Output**



* **%e**

1. **int** main()
2. {
3. **float** y=3;
4. printf("Exponential value of y is: %e", y);
5. **return** 0;
6. }

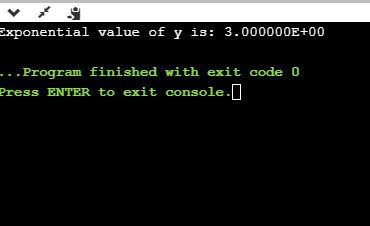
**Output**



* **%E**

1. **int** main()
2. {
3. **float** y=3;
4. printf("Exponential value of y is: %E", y);
5. **return** 0;
6. }

**Output**

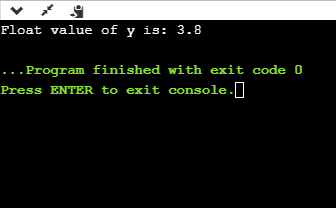


* **%g**

1. **int** main()
2. {
3. **float** y=3.8;
4. printf("Float value of y is: %g", y);
5. **return** 0;
6. }

In the above code, we are displaying the floating value of y by using %g specifier. The %g specifier displays the output same as the input with a same precision.

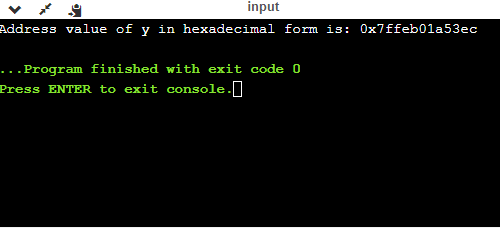
**Output**



* **%p**

1. **int** main()
2. {
3. **int** y=5;
4. printf("Address value of y in hexadecimal form is: %p", &y);
5. **return** 0;
6. }

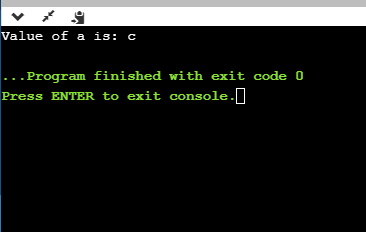
**Output**



* **%c**

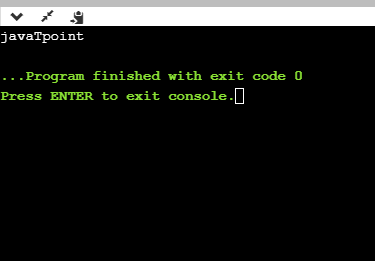
1. **int** main()
2. {
3. **char** a='c';
4. printf("Value of a is: %c", a);
5. **return** 0;
6. }

**Output**



* **%s**

1. **int** main()
2. {
3. printf("%s", "javaTpoint");
4. **return** 0;
5. }

**Output:-**

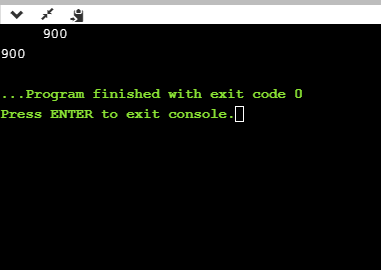
## Minimum Field Width Specifier

Suppose we want to display an output that occupies a minimum number of spaces on the screen. You can achieve this by displaying an integer number after the percent sign of the format specifier.

1. **int** main()
2. {
3. **int** x=900;
4. printf("%8d", x);
5. printf("\n%-8d",x);
6. **return** 0;
7. }

In the above program, %8d specifier displays the value after 8 spaces while %-8d specifier will make a value left-aligned.

**Output**

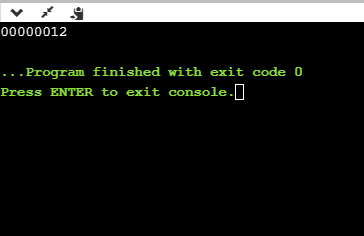


**Now we will see how to fill the empty spaces. It is shown in the below code:**

1. **int** main()
2. {
3. **int** x=12;
4. printf("%08d", x);
5. **return** 0;
6. }

In the above program, %08d means that the empty space is filled with zeroes.

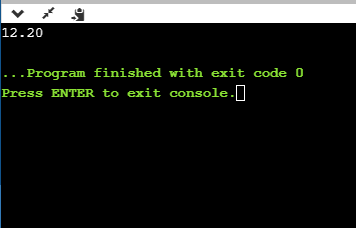
**Output**



## Specifying Precision

We can specify the precision by using '.' (Dot) operator which is followed by integer and format specifier.

1. **int** main()
2. {
3. **float** x=12.2;
4. printf("%.2f", x);
5. **return** 0;
6. }

**Output**

# Escape Sequence in C

An escape sequence in C language is a sequence of characters that doesn't represent itself when used inside string literal or character.

It is composed of two or more characters starting with backslash \. For example: \n represents new line.

## List of Escape Sequences in C

|  |  |
| --- | --- |
| **Escape Sequence** | **Meaning** |
| \a | Alarm or Beep |
| \b | Backspace |
| \f | Form Feed |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab (Horizontal) |
| \v | Vertical Tab |
| \\ | Backslash |
| \' | Single Quote |
| \" | Double Quote |
| \? | Question Mark |
| \nnn | octal number |
| \xhh | hexadecimal number |
| \0 | Null |

## Escape Sequence Example

1. #include<stdio.h>
2. **int** main(){
3. **int** number=50;
4. printf("You\nare\nlearning\n\'c\' language\n\"Do you know C language\"");
5. **return** 0;
6. }

**Output:**

You

are

learning

'c' language

"Do you know C language"

## What is ASCII code?

The full form of ASCII is the **American Standard Code for information interchange**. It is a character encoding scheme used for electronics communication. Each character or a special character is represented by some ASCII code, and each ascii code occupies 7 bits in memory.

In [C programming language](https://www.javatpoint.com/c-programming-language-tutorial), a character variable does not contain a character value itself rather the ascii value of the character variable. The ascii value represents the character variable in numbers, and each character variable is assigned with some number range from 0 to 127. For example, the ascii value of 'A' is 65.