

C Programming

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, strings, functions, file handling, etc. that are being used in many languages like C++, Java, C#, 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.

C Program

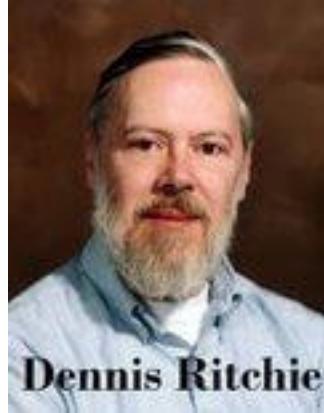
In this tutorial, all C programs are given with C compiler so that you can quickly change the C program code.

File: main.c

```
#include <stdio.h>
int main() {
```

```
printf("Hello C Programming\n");
return 0;
}
```

History of C Language



Dennis Ritchie

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.

Features of C Language

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

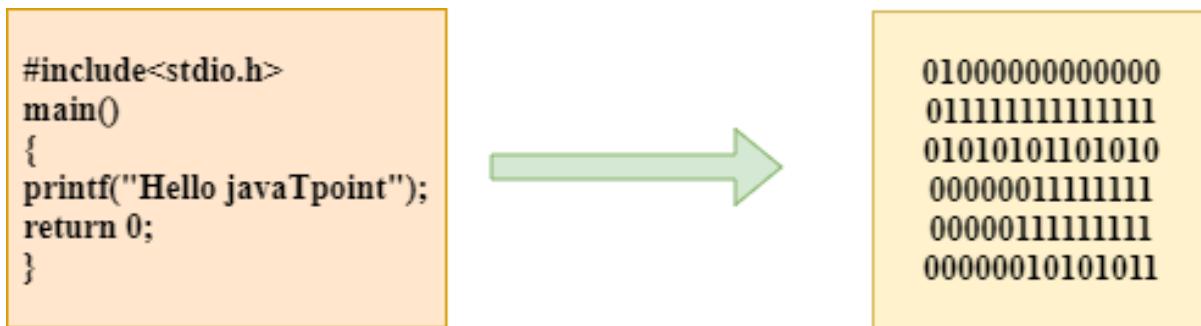
1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed

8. Pointers
9. Recursion
10. Extensible

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.

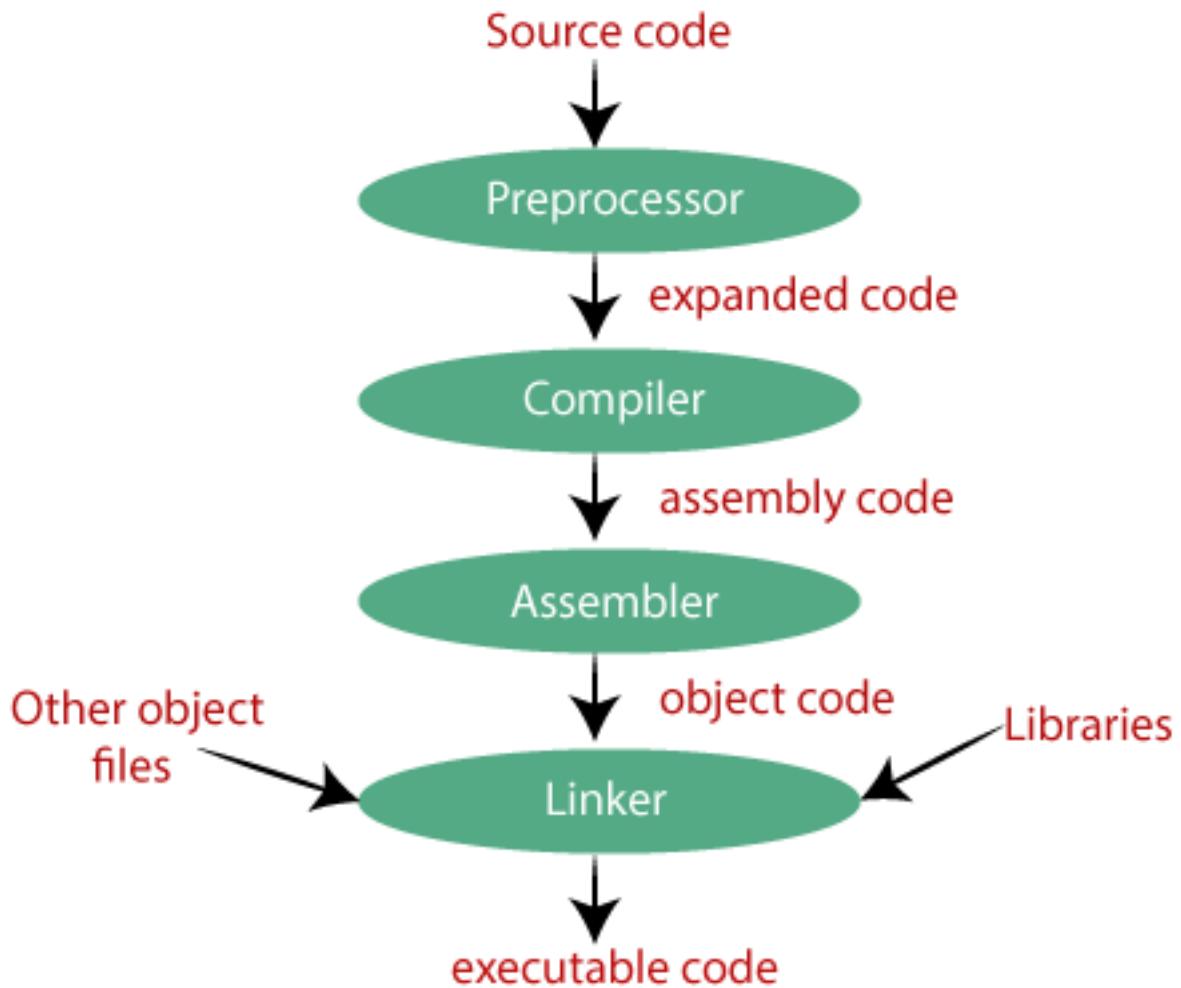


The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps, i.e., Pre-processing, Compiling, Assembling, and Linking.

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:

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



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:

1. `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.

1. `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.

```
#include<stdio.h>
int main(){
    int number;
    printf("enter a number:");
    scanf("%d",&number);
    printf("cube of number is:%d ",number*number*number);
    return 0;
}
```

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.

Program to print sum of 2 numbers

Let's see a simple example of input and output in C language that prints addition of 2 numbers.

```
#include<stdio.h>
int main(){
    int x=0,y=0,result=0;

    printf("enter first number:");
    scanf("%d",&x);
    printf("enter second number:");
    scanf("%d",&y);
    result=x+y;
    printf("sum of %d and %d is %d",x,y,result);
}
```

```

scanf("%d",&x);
printf("enter second number:");
scanf("%d",&y);

result=x+y;
printf("sum of 2 numbers:%d ",result);

return 0;
}

```

Output

```

enter first number:9
enter second number:9
sum of 2 numbers:18

```

Advantages:

1. Efficiency: C is a fast and efficient language that can be used to create high-performance applications.
2. Portability: C programs can be compiled and run on a wide range of platforms and operating systems.
3. Low-level access: C provides low-level access to system resources, making it ideal for systems programming and developing operating systems.
4. Large user community: C has a large and active user community, which means there are many resources and libraries available for developers.
5. Widely used: C is a widely used language, and many modern programming languages are built on top of it.

Disadvantages:

1. Steep learning curve: C can be difficult to learn, especially for beginners, due to its complex syntax and low-level access to system resources.
2. Lack of memory management: C does not provide automatic memory management, which can lead to memory leaks and other memory-related bugs if not handled properly.
3. No built-in support for object-oriented programming: C does not provide built-in support for object-oriented programming, making it more difficult to write object-oriented code compared to languages like Java or Python.

4. No built-in support for concurrency: C does not provide built-in support for concurrency, making it more difficult to write multithreaded applications compared to languages like Java or Go.
5. Security vulnerabilities: C programs are prone to security vulnerabilities, such as buffer overflows, if not written carefully. Overall, C is a powerful language with many advantages, but it also requires a high degree of expertise to use effectively and has some potential drawbacks, especially for beginners or developers working on complex projects.

C Token – Keywords

The [keywords](#) are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program. Since keywords are referred names for a compiler, they can't be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed. You cannot redefine keywords. However, you can specify the text to be substituted for keywords before compilation by using C preprocessor directives. C language supports **32** keywords which are given below:

```
auto    double   int     struct
break   else     long    switch
case    enum     register  typedef
char    extern   return   union
const   float    short   unsigned
continue  for    signed   void
default  goto    sizeof   volatile
do      if     static   while
```

C Token – Identifiers

Identifiers are used as the general terminology for the naming of variables, functions, and arrays. These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character. Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use. Once declared, you can use the identifier in later program statements to refer to the associated value. A special identifier called a statement label can be used in goto statements.