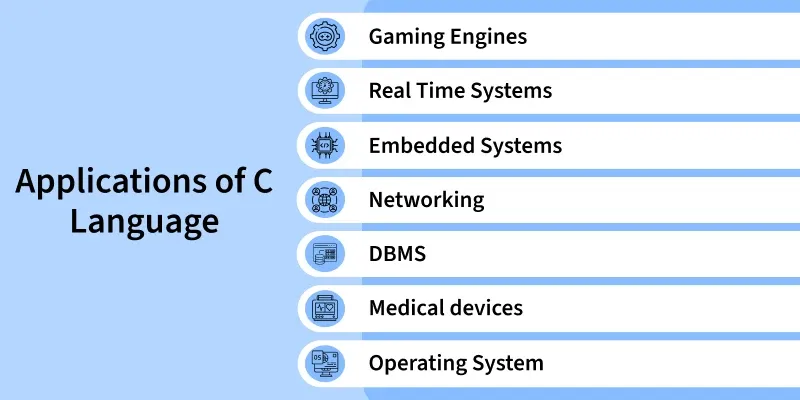
**The basic of c language**

C is a general-purpose mid-level programming language developed by Dennis M. Ritchie at Bell Laboratories in 1972. It was initially used for the development of UNIX operating system, but it later became popular for a wide range of applications. Today, C remains one of the top three most widely used programming languages.



**C Hello World Program**

The “Hello World” program is the first step towards learning any programming language. It is also one of the simplest programs that is used to introduce aspiring programmers to the programming language. It typically outputs the text "Hello, World!" to the console screen.

C Program to Print "Hello World"

To print the “Hello World”, we can use the [printf function](https://www.geeksforgeeks.org/c/printf-in-c/" \t "_blank) from the stdio.h library that prints the given string on the screen. Provide the string "Hello World" to this function as shown in the below code:

#include <stdio.h>

// Main function: entry point for execution

int main() {

// Writing print statement to print hello world

printf("Hello World");

return 0;

}

**Output**

Hello World

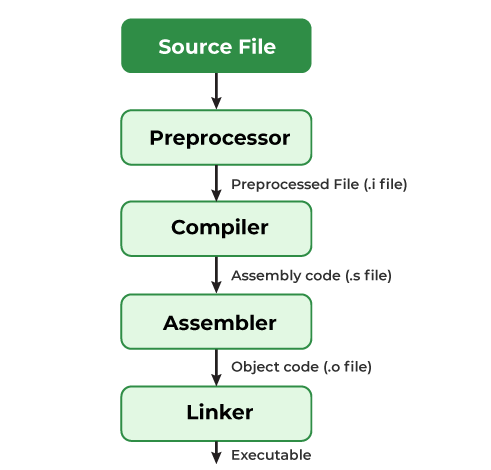
**Explanation:**

* **#include <stdio.h>** – This line includes the standard input-output library in the program.
* **int main()**– The main function where the execution of the program begins.
* **printf(“Hello, World!\n”);** – This function call prints “Hello, World!” followed by a new line.
* **return 0;**-This statement indicates that the program ended successfully.

**Compiling a C Program: Behind the Scenes**

The compilation is the process of converting the source code of the C language into machine code. As C is a mid-level language, it needs a compiler to convert it into an executable code so that the program can be run on our machine.

The C program goes through the following phases during compilation:



**C Comments**

The **comments in C** are human-readable notes in the source code of a C program used to make the program easier to read and understand. They are not a part of the executable program by the compiler or an interpreter.

#include <stdio.h>

int main() {

// This is a comment, the below

// statement will not be executed

// printf("Hi from comment");

printf("Hello");

return 0;

}

**Output**

Hello

**Types of Comments in C**

In C, there are two types of comments in C language:

* **Single-line Comments**
* **Multi-line Comments**

**Single-line Comments**

Single-line comments are used to comment out a single line of code or a part of it. The single line comments in C start with **two forward slashes (//)**, and everything after the slashes on that line is considered a comment. They are also called [C++ Style comments](https://www.geeksforgeeks.org/cpp/cpp-comments/) as they were first introduced in C++ and later adopted in C also.

**Syntax**

*// This is a single line comment*

**Example:**

#include <stdio.h>

​

int main() {

// This is a single-line comment explaining the variable x

int x = 5;

// Output the value of x

printf("Value of x: %d\n", x);

return 0;

}

**Output**

Value of x: 5

In this C program, the comments provide explanations about the code. The compiler ignores the comments and does not execute them.

We can also create a comment that displays at the end of a line of code using a single-line comment. But generally, it's better to practice putting the comment before the line of code.

#include <stdio.h>

​

int main() {

// single line comment here

printf("Hi"); // After line comment here

return 0;

}

**Output**

Hi

**Multi-line Comments**

Multi-line comments in C are used write comments that span **more than one line.**They are generally used for longer descriptions or for commenting out multiple lines of code. In C language, these comments**begin with /\*** and **end with \*/.**Everything between these markers is treated as a comment.

**Syntax:**

*/\* This is a multi-line comment*

*which can span multiple lines \*/*

**Example:**

#include <stdio.h>

​

int main() {

/\*

This comment contains some code which

will not be executed.

printf("Code enclosed in Comment");

\*/

printf("Welcome to GeeksforGeeks");

return 0;

}

**Output**

Program runs without errors.

**Tokens in C**

In C programming, tokens are the smallest units in a program that have meaningful representations. Tokens are the building blocks of a C program, and they are recognized by the C compiler to form valid expressions and statements. Tokens can be classified into various categories, each with specific roles in the program.



The tokens of C language can be classified into six types based on the functions they are used to perform. The types of C tokens are as follows:

**Table of Content**

* [Punctuators](https://www.geeksforgeeks.org/c/tokens-in-c/#1-punctuators)
* [Keywords](https://www.geeksforgeeks.org/c/tokens-in-c/#keywords)
* [Strings](https://www.geeksforgeeks.org/c/tokens-in-c/#3-strings)
* [Operators](https://www.geeksforgeeks.org/c/tokens-in-c/#4-operators)
* [Identifiers](https://www.geeksforgeeks.org/c/tokens-in-c/#identifiers)
* [Constants](https://www.geeksforgeeks.org/c/tokens-in-c/#constants)

**1. Punctuators**

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose. Some of these are listed below:

* **Brackets[]:** Opening and closing brackets are used as array element references. These indicate single and multidimensional subscripts.
* **Parentheses():**These special symbols are used to indicate function calls and function parameters.
* **Braces{}:** These opening and ending curly braces mark the start and end of a block of code containing more than one executable statement.
* **Comma (, ):**It is used to separate more than one statement like for separating parameters in function calls.
* **Colon(:):** It is an operator that essentially invokes something called an initialization list.
* **Semicolon(;):** It is known as a statement terminator.  It indicates the end of one logical entity. That's why each individual statement must be ended with a semicolon.
* **Asterisk (\*):**It is used to create a pointer variable and for the multiplication of variables.
* **Assignment operator(=):**It is used to assign values and for logical operation validation.
* **Pre-processor (#):**The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.
* **Dot (.):** Used to access members of a structure or union.
* **Tilde(~):**Bitwise One's Complement Operator.

**Example:**

#include <stdio.h>

​

int main() {

// '\n' is a special symbol that

// represents a newline

printf("Hello, \n World!");

return 0;

}

**Output**

Hello,

World!

**2. Keywords**

[**Keywords**](https://www.geeksforgeeks.org/c/keywords-in-c/)are reserved words that have predefined meanings in C. These cannot be used as identifiers (variable names, function names, etc.). Keywords define the structure and behavior of the program **C** language supports **32** keywords such as int, for, if, ... etc.

**Example:**

#include <stdio.h>

​

int main() {

// 'int' is a keyword used to define

// variable type

int x = 5;

printf("%d", x);

// 'return' is a keyword used to exit

// main function

return 0;

}

**Output**

5

***Note:*** *The number of keywords may change depending on the version of C you are using. For example, keywords present in ANSI C are 32 while in C11, it was increased to 44. Moreover, in the latest c23, it is increased to around 54.*

**3. Strings**

[**Strings**](https://www.geeksforgeeks.org/c/strings-in-c/) are nothing but an array of characters ended with a null character (‘\0’). This null character indicates the end of the string. Strings are always enclosed in double quotes. Whereas a character is enclosed in single quotes in C and C++.

**Examples:**

#include <stdio.h>

​

int main() {

// "Hello, World!" is a string literal

char str[] = "Hello, World!";

printf("%s", str);

return 0;

}

**Output**

Hello, World!

**4. Operators**

[Operators](https://www.geeksforgeeks.org/c/operators-in-c/) are symbols that trigger an action when applied to C variables and other objects. The data items on which operators act are called operands.

**Example:**

#include <stdio.h>

​

int main() {

int a = 10, b = 5;

// '+' is an arithmetic operator used

// for addition

int sum = a + b;

printf("%d", sum);

return 0;

}

**Output**

15

**5. Identifiers**

[**Identifiers**](https://www.geeksforgeeks.org/c/c-identifiers/) are names given to variables, functions, arrays, and other user-defined items. They must begin with a letter (a-z, A-Z) or an underscore (\_) and can be followed by letters, digits (0-9), and underscores.

**Example:**

#include <stdio.h>

​

int main() {

// 'num' is an identifier used to name

// a variable

int num = 10;

printf("%d", num);

return 0;

}

**Output**

10

**6. Constants**

[**Constants**](https://www.geeksforgeeks.org/c/constants-in-c/) are fixed values used in a C program. These values do not change during the execution of the program. Constants can be integers, floating-point numbers, characters, or strings.

**Examples:**

#include <stdio.h>

​

int main() {

// 'MAX\_VALUE' is a constant that holds

// a fixed value

const int MAX\_VALUE = 100;

printf("%d", MAX\_VALUE);

return 0;

}

**Output**

100

**Output**

Welcome to GeeksforGeeks

**Nesting Comments in C**

In C language, comments cannot be nested. That means you cannot place a multi-line comment inside another multi-line comment in C language. If you try to do so, the compiler will treat the closing \*/ of the inner comment as the end of the entire multi-line comment. and the rest of the part after the **inner closing (\*/)**will not be commented.

#include <stdio.h>

​

int main() {

/\* This is the start of a multi-line comment

/\* This is an inner multi-line comment \*/

This line will cause an error because the compiler

considers the above '\*/' as the end of the comment block.

\*/

printf("This program will not compile.\n");

return 0;

}

**Output**

./Solution.c: In function 'main':  
./Solution.c:6:8: error: unknown type name 'This'  
 This line will cause an error because the compiler  
 ^  
./Solution.c:6:18: error: expected '=', ',', ';', 'asm' or '\_\_attribute\_\_' before 'will'  
 This line will cause an error because the compiler  
 ^

If nested comments are needed, it’s best to **use single-line comments**or comment out individual parts.

#include <stdio.h>

​

int main() {

// /\* This block of code is commented out

// int x = 10;

// printf("Value of x: %d\n", x);

// \*/

​

printf("Program runs without errors.\n");

return 0;

}