**C Language - Overview**

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 (Digital Equipment Corporation Programmed Data Processor-11 ) computer in 1972.

In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.

The UNIX operating system, the C compiler, and essentially all UNIX application programs have been written in C. C has now become a widely used professional language for various reasons −

* Easy to learn
* Structured language
* It produces efficient programs
* It can handle low-level activities
* It can be compiled on a variety of computer platforms

**Facts about C**

* C was invented to write an operating system called UNIX.
* C is a successor of B language which was introduced around the early 1970s.
* The language was formalized in 1988 by the American National Standard Institute (ANSI).
* The UNIX OS was totally written in C.
* Today C is the most widely used and popular System Programming Language.
* Most of the state-of-the-art software have been implemented using C.
* Today's most popular Linux OS and RDBMS MySQL have been written in C.

## Applications of C Programming

C was initially used for system development work, particularly the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of the use of C might be −

* Operating Systems
* Language Compilers
* Assemblers
* Text Editors
* Print Spoolers
* Network Drivers
* Modern Programs
* Databases
* Language Interpreters
* Utilities

**C Programs**

A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension **".c"**; for example, *hello.c*. You can use **"vi"**, **"vim"** or any other text editor to write your C program into a file. You can also use Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language.

**C - Program Structure**

Before we study the basic building blocks of the C programming language, let us look at a bare minimum C program structure so that we can take it as a reference in the upcoming chapters.

**Hello World Example**

A C program basically consists of the following parts −

* Preprocessor Commands
* Functions
* Variables
* Statements & Expressions
* Comments

Let us look at a simple code that would print the words "Hello World" −

#include <stdio.h>

int main() {

/\* my first program in C \*/

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

return 0;

}

Let us take a look at the various parts of the above program −

* The first line of the program *#include <stdio.h>* is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
* The next line *int main()* is the main function where the program execution begins.
* The next line /\*...\*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
* The next line *printf(...)* is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
* The next line **return 0;** terminates the main() function and returns the value 0.

**Compile and Execute C Program**

Let us see how to save the source code in a file, and how to compile and run it. Following are the simple steps −

* Open a text editor and add the above-mentioned code.
* Save the file as *hello.c*
* Open a command prompt and go to the directory where you have saved the file.
* Type *gcc hello.c* and press enter to compile your code.
* If there are no errors in your code, the command prompt will take you to the next line and would generate *a.out* executable file.
* Now, type *a.out* to execute your program.
* You will see the output *"Hello World"* printed on the screen.

$ gcc hello.c

$ ./a.out

Hello, World!

Make sure the gcc compiler is in your path and that you are running it in the directory containing the source file hello.c.

# C - Basic Syntax

You have seen the basic structure of a C program, so it will be easy to understand other basic building blocks of the C programming language.

**Character set**

A character set is a set of alphabets, letters and some special characters that are valid in C language.

**Alphabets**

Uppercase: A B C ................................... X Y Z

Lowercase: a b c ...................................... x y z

C accepts both lowercase and uppercase alphabets as variables and functions.

**Digits**

0 1 2 3 4 5 6 7 8 9

**Special Characters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Special Characters in C Programming | | | | |
| , | < | > | . | \_ |
| ( | ) | ; | $ | : |
| % | [ | ] | # | ? |
| ' | & | { | } | " |
| ^ | ! | \* | / | | |
| - | \ | ~ | + |  |

**White space Characters**

Blank space, newline, horizontal tab, carriage, return and form feed.

## Tokens in C

A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following C statement consists of five tokens −

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

The individual tokens are −

printf

(

"Hello, World! \n"

)

;

## Semicolons

In a C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

Given below are two different statements −

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

return 0;

## Comments

Comments are like helping text in your C program and they are ignored by the compiler. They start with /\* and terminate with the characters \*/ as shown below −

/\* my first program in C \*/

You cannot have comments within comments and they do not occur within a string or character literals.

## Identifiers

A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z, a to z, or an underscore '\_' followed by zero or more letters, underscores, and digits (0 to 9).

C does not allow punctuation characters such as @, $, and % within identifiers. C is a **case-sensitive** programming language. Thus, *Manpower* and *manpower* are two different identifiers in C. Here are some examples of acceptable identifiers −

mohd zara abc move\_name a\_123

myname50 \_temp j a23b9 retVal

**Rules for naming identifiers**

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
2. The first letter of an identifier should be either a letter or an underscore.
3. You cannot use keywords as identifiers.
4. There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than 31 characters.

You can choose any name as an identifier if you follow the above rule, however, give meaningful names to identifiers that make sense.

## Keywords

The following list shows the reserved words in C. These reserved words may not be used as constants or variables or any other identifier names.

|  |  |  |  |
| --- | --- | --- | --- |
| auto | else | long | Switch |
| break | enum | register | Typedef |
| case | extern | return | Union |
| char | float | short | Unsigned |
| const | for | signed | Void |
| continue | goto | sizeof | volatile |
| default | if | static | While |
| do | int | struct | \_Packed |
| double |  |  |  |

## Whitespace in C

A line containing only whitespace, possibly with a comment, is known as a blank line, and a C compiler totally ignores it.

Whitespace is the term used in C to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the following statement −

int age;

there must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them. On the other hand, in the following statement −

fruit = apples + oranges; // get the total fruit

no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish to increase readability.