C Tutorial

C language is considered as the mother language of all programming languages. It is a powerful general-purpose programming language. It can be used to develop software like database, OS, compiler, etc.

Get Started

Welcome to the C Tutorial tutorial series! To begin learning:

1. Select a lesson from the sidebar on the left
2. Work through the lessons in order for the best learning experience
3. Each lesson contains detailed explanations and examples

C Overview

What is C?

- Since the late 19th century, C has been a popular programming language for general-purpose use.
- C language was developed by Dennis M. Ritchie at Bell Laboratories in the early 1970s.
- Its applications are very diverse. It ranges from developing operating systems to databases and more. It is a system programming language used to do low-level programming (e.g., driver or kernel).
- Even if it’s old, it is still a very popular programming language.
- As the whole UNIX operating system was written in C, it has a strong association with operating systems.
- C has also been used widely while creating iOS and Android kernels.
- MySQL database is written using C.
- Ruby and Perl are mostly written using C.
- Most parts of Apache and NGINX are written using C.
- Embedded Systems are created using C.

Why should we learn C/ Features of C?

- As mentioned above, it is one of the most popular programming languages in the world.

- Learning any other popular programming language such as Python or C++ becomes much easier if you know C.
- C is a flexible language, proven by the fact that it can be used in a variety of applications as well as technologies.
- C is very fast when compared to other programming languages, be it Java or Python.
- C takes only significant CPU time for interpretation. That is why a lot of Python libraries such as NumPy, pandas, Scikit-learn, etc., are built using C.
- Being close to machine language, some of its functions include direct access to machine-level hardware APIs.
- It is a structural language (follows a specific structure) and a compiled language.
- It is a procedural programming language (POP). Procedural programming is the use of code in a step-wise procedure to develop applications.

How is it different from C++?

- The syntax of C++ is almost identical to that of C, as C++ was developed as an extension of C.
- In contrast to C, C++ supports classes and objects, while C does not.
- C gives most of the control to the hands of users. Things like memory allocation and manipulation are totally in the hands of the programmer. Being a flexible language, it provides more access to the programmer, making it more efficient.
- C is POP (procedure-oriented programming), whereas C++ is OOP (object-oriented programming).

Getting Started with C

Requirements before you start

- To start using C, you need two things:
- A text editor, like Notepad, or an IDE, like VSCode to act as a platform for you to write C code.
- A compiler, like GCC to translate the C code you have written, which is a high-level language, into a low-level language that the computer will understand.

What is an IDE?

- IDE stands for Integrated Development Environment.

- It is nothing more than an enhanced version of a text editor that helps you write more efficient and nicer code.
- It helps to differentiate different parts of your code with different colors and notifies you if you are missing some semicolon or bracket at some place by highlighting that area.
- A lot of IDEs are available, such as DEVC++ or Code Blocks, but we will prefer using VS Code for this tutorial series.

Installing VSCode

- Visit https://code.visualstudio.com/download
- Click on the download option as per your operating system.
- After the download is completed, open the setup and run it by saving VS Code in the default location without changing any settings.
- You will need to click the next button again and again until the installation process begins.

What is a Compiler?

- A compiler is used to run the program of a certain language, which is generally high-level, by converting the code into a language that is low-level that our computer could understand.
- There are a lot of compilers available, but we will proceed with teaching you to use MinGW for this course because it will fulfill all of our requirements, and also it is recommended by Microsoft itself.

Setting up the compiler

- Visit https://code.visualstudio.com/docs/languages/cpp
- Select C++ from the sidebar.
- Choose “GCC via Mingw-w64 on Windows” from the options shown there.
- Select the install sourceforge option.
- After the download gets completed, run the setup and choose all the default options as we did while installing VS Code.

Setting Path for Compiler

- Go to the C directory. Navigate into the Program Files. Then, open MinGW-64. Open MinGW-32. And then the bin folder. After reaching the bin, save the path or URL to the bin.

- Then go to the properties of ‘This PC’.
- Select ‘Advance System Settings’.
- Select the ‘Environment Variable’ option.
- Add the copied path to the Environment Variable.
- And now, you can visit your IDE and run your C programs on it. The configuration part is done.

Basic Structure & Syntax

Programming in C involves following a basic structure throughout. Here’s what it can be broken down to.

- Pre-processor commands
- Functions
- Variables
- Statements
- Expressions
- Comments

Pre-processor commands

Pre-processor commands are commands which tell our program that before its execution, it must include the file name mentioned in it because we are using some of the commands or codes from this file.

They add functionalities to a program.

One example could be,

```
#include <math.h>
```

We include math.h to be able to use some special functions like power and absolute. #include<filename.h> is how we include them into our programs.

Detailed explanations of everything else in the structure will follow in the later part of the tutorial.

Header files:

- Collection of predefined/built-in functions developed
- It is always declared on the heading side of the program hence it is called a header file

- It is identified with the extension (.h)
- It gets installed while installing IDE (integrated development environment)
- It stores functions as per their categories hence they are called library

Syntax

An example below shows how a basic C program is written.

Declaration of header file //name of the header files of which functions are been usedmain() /*it is called main function which stores the execution of program*/{ //start of the program //program statements} //end of the program

- Here, the first line is a pre-processor command including a header file stdio.h.
- C ignores empty lines and spaces.
- There is a main() function then, which should always be there.

A C program is **made up of different tokens combined**. These tokens include:

- Keywords
- Identifiers
- Constants
- String Literal
- Symbols

Keywords

Keywords are reserved words that cannot be used elsewhere in the program for naming a variable or a function. They have a specific function or task and they are solely used for that. Their functionalities are pre-defined.

One such example of a keyword could be return which is used to build return statements for functions. Other examples are auto, if, default, etc.

Whenever we write any keyword in IDE their color slightly changes and it looks different from other variables or functions. For example, in Turbo C all keywords turn into white color.

Identifiers

Identifiers are names given to variables or functions to differentiate them from one another. Their definitions are solely based on our choice but there are a few rules that we have to follow while naming identifiers. One such rule says that the name cannot contain special symbols such as @, -, *, <, etc.

C is a case-sensitive language so an identifier containing a capital letter and another one containing a small letter in the same place will be different. For example, the three words: Code, code, and cOde can be used as three different identifiers.

Rules for naming identifiers:

1. One should not name any identifier starting with a numeric value or symbol. It should start only with an underscore or alphabet.
2. They should not contain spaces.
3. Giving logical names is recommended as per our program.

Constants

Constants are very similar to a variable and they can also be of any data type. The only difference between a constant and a variable is that a constant's value never changes. We will see constants in more detail in the upcoming tutorial.

String Literal

String literals or string constants are a sequence of characters enclosed in double quotation marks. For example, "This is a string literal!" is a string literal. C method printf() utilizes the same to format the output.

C Comments

Comments can be used to insert any informative piece which a programmer does not wish to be executed. It could be either to explain a piece of code or to make it more readable. In addition, it can be used to prevent the execution of alternative code when the process of debugging is done.

Comments can be single-lined or multi-lined.

Single Line Comments

- Single-line comments start with two forward slashes (//).
- Any information after the slashes // lying on the same line would be ignored (will not be executed).

An example of how we use a single-line comment:

```
#include <stdio.h>


int main()
```

```
{
  // This is a single line comment
  printf("Hello World!");
  return 0;
}
```

Multi-line Comments

- A multi-line comment starts with /* and ends with */.
- Any information between /* and */ will be ignored by the compiler.

An example of how we use a multi-line comment:

```
#include <stdio.h>


int main()
{
  /* This is a
  multi-line
  comment */
  printf("Hello World!");
  return 0;
}
```

C Variables

Variables are **containers for storing data values.**

In C, there are different types of variables.

For example:

- An integer variable defined with the keyword int stores integers (whole numbers), without decimals, such as 91 or -13.
- A floating point variable defined with the keyword float stores floating point numbers, with decimals, such as 99.98 or -1.23.
- A character variable defined with the keyword char stores single characters, such as 'A' or 'z'. Char values are bound to be surrounded by single quotes.

Declaration

We cannot declare a variable without specifying its data type. The data type of a variable depends on what we want to store in the variable and how much space we want it to hold. The syntax for declaring a variable is simple:

data_type variable_name;

OR

data_type variable_name = value;

Naming a Variable

There is no limit to what we can call a variable. Yet there are specific rules we must follow while naming a variable:

- A variable name can only contain alphabets, digits, and underscores (_).
- A variable cannot start with a digit.
- A variable cannot include any white space in its name.
- The name should not be a reserved keyword or any special character.

A variable, as its name is defined, can be altered, or its value can be changed, but the same is not true for its type. If a variable is of integer type, then it will only store an integer value through a program. We cannot assign a character type value to an integer variable. We cannot even store a decimal value into an integer variable.