



LEARN C PROGRAMMING

c programming language

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About The Tutorial

C is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system.

C is the most widely used computer language. It keeps fluctuating at number one scale of popularity along with Java programming language, which is also equally popular and most widely used among modern software programmers.

Audience

This tutorial is designed for software programmers with a need to understand the C programming language starting from scratch. This tutorial will give you enough understanding on C programming language from where you can take yourself to higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages will help you in understanding the C programming concepts and move fast on the learning track.

Copyright & Disclaimer

© Copyright 2014 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About The Tutorial	i
Audience	i
Prerequisites	i
Copyright & Disclaimer	i
Table of Contents	ii
1. OVERVIEW	1
Facts about C	1
Why Use C?	1
C Programs	2
2. ENVIRONMENT SETUP	3
Try it Option Online	3
Local Environment Setup	3
Text Editor	3
The C Compiler	4
Installation on UNIX/Linux	4
Installation on Mac OS	5
Installation on Windows	5
3. PROGRAM STRUCTURE	6
Hello World Example	6
Compile and Execute C Program	7
4. BASIC SYNTAX	8
Tokens in C	8
Semicolons	8
Comments	8
Identifiers	9

Keywords	9
Whitespace in C	10
5. DATA TYPES.....	11
Integer Types	11
Floating-Point Types	13
The void Type.....	14
6. VARIABLES.....	15
Variable Definition in C	15
Variable Declaration in C.....	16
Lvalues and Rvalues in C	18
7. CONSTANTS AND LITERALS	19
Integer Literals	19
Floating-point Literals	20
Character Constants.....	20
String Literals	21
Defining Constants.....	22
The #define Preprocessor	22
The const Keyword	23
8. STORAGE CLASSES.....	24
The auto Storage Class	24
The register Storage Class	24
The static Storage Class.....	25
The extern Storage Class	26
9. OPERATORS.....	28
Arithmetic Operators	28
Relational Operators.....	30

Logical Operators	32
Bitwise Operators	34
Assignment Operators	37
Misc Operators \rightarrow sizeof & ternary	40
Operators Precedence in C.....	41
10. DECISION MAKING	45
if Statement	46
if...else Statement	48
if...else if...else Statement	49
Nested if Statements	51
switch Statement	53
Nested switch Statements	55
The ? : Operator:.....	57
11. LOOPS	58
while Loop	59
for Loop	61
do...while Loop	63
Nested Loops	65
Loop Control Statements	67
break Statement	68
continue Statement	70
goto Statement.....	72
The Infinite Loop.....	74
12. FUNCTIONS	76
Defining a Function.....	76
Function Declarations	77

Calling a Function.....	78
Function Arguments.....	79
Call by Value	80
Call by Reference	81
13. SCOPE RULES.....	84
Local Variables	84
Global Variables.....	85
Formal Parameters	86
Initializing Local and Global Variables	87
14. ARRAYS	89
Declaring Arrays.....	89
Initializing Arrays	89
Accessing Array Elements	90
Arrays in Detail	91
Multidimensional Arrays	92
Two-dimensional Arrays.....	92
Initializing Two-Dimensional Arrays	93
Accessing Two-Dimensional Array Elements.....	93
Passing Arrays to Functions.....	94
Return Array from a Function	96
Pointer to an Array	99
15. POINTERS.....	101
What are Pointers?	101
How to Use Pointers?.....	102
NULL Pointers	103
Pointers in Detail	104
Pointer Arithmetic.....	104
Incrementing a Pointer.....	105

Decrementing a Pointer	106
Pointer Comparisons	107
Array of Pointers	108
Pointer to Pointer	110
Passing Pointers to Functions	112
Return Pointer from Functions	114
16. STRINGS	117
17. STRUCTURES	120
Defining a Structure	120
Accessing Structure Members	121
Structures as Function Arguments	122
Pointers to Structures	124
Bit Fields	126
18. UNIONS	128
Defining a Union	128
Accessing Union Members	129
19. BIT FIELDS	132
Bit Field Declaration	133
20. TYPEDEF	136
typedef vs #define	137
21. INPUT AND OUTPUT	139
The Standard Files	139
The getchar() and putchar() Functions	139
The gets() and puts() Functions	140
The scanf() and printf() Functions	141
22. FILE I/O	143

Opening Files	143
Closing a File	144
Writing a File.....	144
Reading a File.....	145
Binary I/O Functions	146
23. PREPROCESSORS	147
Preprocessors Examples.....	148
Predefined Macros.....	148
Preprocessor Operators	150
The Macro Continuation (\) Operator	150
The Stringize (#) Operator	150
The Token Pasting (##) Operator	150
The Defined() Operator	151
Parameterized Macros	152
24. HEADER FILES.....	153
Include Syntax.....	153
Include Operation	153
Once-Only Headers	154
Computed Includes	155
25. TYPE CASTING	156
Integer Promotion	157
Usual Arithmetic Conversion.....	157
26. ERROR HANDLING	160
errno, perror(), and strerror()	160
Divide by Zero Errors.....	161
Program Exit Status.....	162

27. RECURSION	164
Number Factorial	164
Fibonacci Series	165
28. VARIABLE ARGUMENTS.....	167
29. MEMORY MANAGEMENT	170
Allocating Memory Dynamically	170
Resizing and Releasing Memory	172
30. COMMAND LINE ARGUMENTS.....	174

1. 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 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.

Why Use C?

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.

This tutorial assumes that you know how to edit a text file and how to write source code inside a program file.

2. ENVIRONMENT SETUP

Try it Option Online

You really do not need to set up your own environment to start learning C programming language. Reason is very simple, we already have set up C Programming environment online, so that you can compile and execute all the available examples online at the same time when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try following example using our online compiler option available at <http://www.compileonline.com/>.

```
#include <stdio.h>

int main()
{
    /* my first program in C */
    printf("Hello, World! \n");

    return 0;
}
```

For most of the examples given in this tutorial, you will find the **Try it** option in our website code sections at the top right corner that will take you to the online compiler. So just make use of it and enjoy your learning.

Local Environment Setup

If you want to set up your environment for C programming language, you need the following two software tools available on your computer, (a) Text Editor and (b) The C Compiler.

Text Editor

This will be used to type your program. Examples of a few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

The name and version of text editors can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on Windows as well as on Linux or UNIX.

The files you create with your editor are called the source files and they contain the program source codes. The source files for C programs are typically named with the extension ".c".

Before starting your programming, make sure you have one text editor in place and you have enough experience to write a computer program, save it in a file, compile it and finally execute it.

The C Compiler

The source code written in source file is the human readable source for your program. It needs to be "compiled" into machine language so that your CPU can actually execute the program as per the instructions given.

The compiler compiles the source codes into final executable programs. The most frequently used and free available compiler is the GNU C/C++ compiler, otherwise you can have compilers either from HP or Solaris if you have the respective operating systems.

The following section explains how to install GNU C/C++ compiler on various OS. We keep mentioning C/C++ together because GNU gcc compiler works for both C and C++ programming languages.

Installation on UNIX/Linux

If you are using **Linux or UNIX**, then check whether GCC is installed on your system by entering the following command from the command line:

```
$ gcc -v
```

If you have GNU compiler installed on your machine, then it should print a message as follows:

```
Using built-in specs.
Target: i386-redhat-linux
Configured with: ../configure --prefix=/usr .....
Thread model: posix
gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
```

If GCC is not installed, then you will have to install it yourself using the detailed instructions available at <http://gcc.gnu.org/install/>.

This tutorial has been written based on Linux and all the given examples have been compiled on the Cent OS flavor of the Linux system.

Installation on Mac OS

If you use Mac OS X, the easiest way to obtain GCC is to download the Xcode development environment from Apple's web site and follow the simple installation instructions. Once you have Xcode setup, you will be able to use GNU compiler for C/C++.

Xcode is currently available at developer.apple.com/technologies/tools/.

Installation on Windows

To install GCC on Windows, you need to install MinGW. To install MinGW, go to the MinGW homepage, www.mingw.org, and follow the link to the MinGW download page. Download the latest version of the MinGW installation program, which should be named MinGW-<version>.exe.

While installing MinGW, at a minimum, you must install gcc-core, gcc-g++, binutils, and the MinGW runtime, but you may wish to install more.

Add the bin subdirectory of your MinGW installation to your **PATH** environment variable, so that you can specify these tools on the command line by their simple names.

After the installation is complete, you will be able to run gcc, g++, ar, ranlib, dlltool, and several other GNU tools from the Windows command line.

3. 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:

1. 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.
2. The next line `int main()` is the main function where the program execution begins.

3. 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.
4. The next line `printf(...)` is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
5. The next line **return 0;** terminates the `main()` function and returns the value 0.

End of ebook preview

If you liked what you saw...

Buy it from our store @ <https://store.tutorialspoint.com>