Basic Information About C Language [Updated]

blogwaping.com/2020/07/c-language.html

Do you want to **learn** basic information about the **c Language**?

Yes!

That's great.

This article is the right choice for you.

Here, I will provide you all the basic information about C language.

Introduction Of C Language

C is a high-level computer **programming language**.

It is also known as:

- Mother programming language
- System programming language
- Mid-level programming language
- Procedure-oriented programming language
- Structured programming language

Usually, this language is **designed** to be compiled with a relatively simple compiler.

It provides **low-level** access to memory.

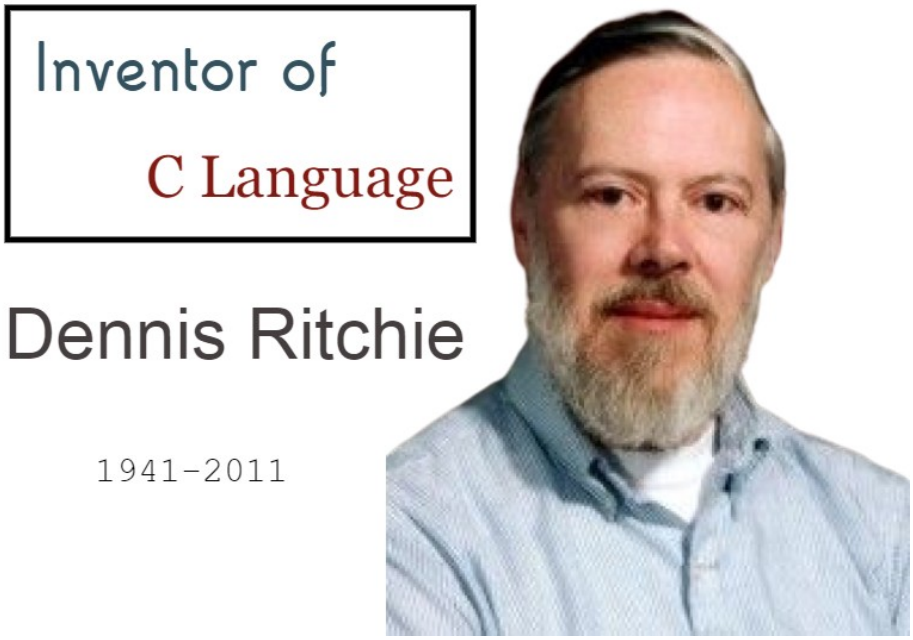
So, it requires **minimum** runtime support to **process** instructions.

If you **learn** this language, another programming language is **easy** to understand for you.

History Of C Language

It is **interesting** to know the history of the C language.

Here, I discuss a **brief history** of the c language.



It was originally invented by **Dennis Ritchie** in **1972** at **AT & T's Bell Laboratory** in the **USA**.

It was primarily developed to writing **UNIX** operating system.

Gradually, it becomes a very **popular** programming language in the worldwide.

It has been standardized by the American National Standards Institute (**ANSI**) since **1989** and subsequently by the International Organization for Standardization (**ISO**).

Timeline of C language development

Version Name	Year	Developer
C	1972	Dennis Ritchie
K&R C	1978	Brian Kernighan & Dennis Ritchie
ANSI C	1989	ANSI Committee
ISO C	1990	ISO Committee
C99	1999	Standardization Committee
C11	2011	Standardization Committee
C18	2017/2018	Standardization Committee

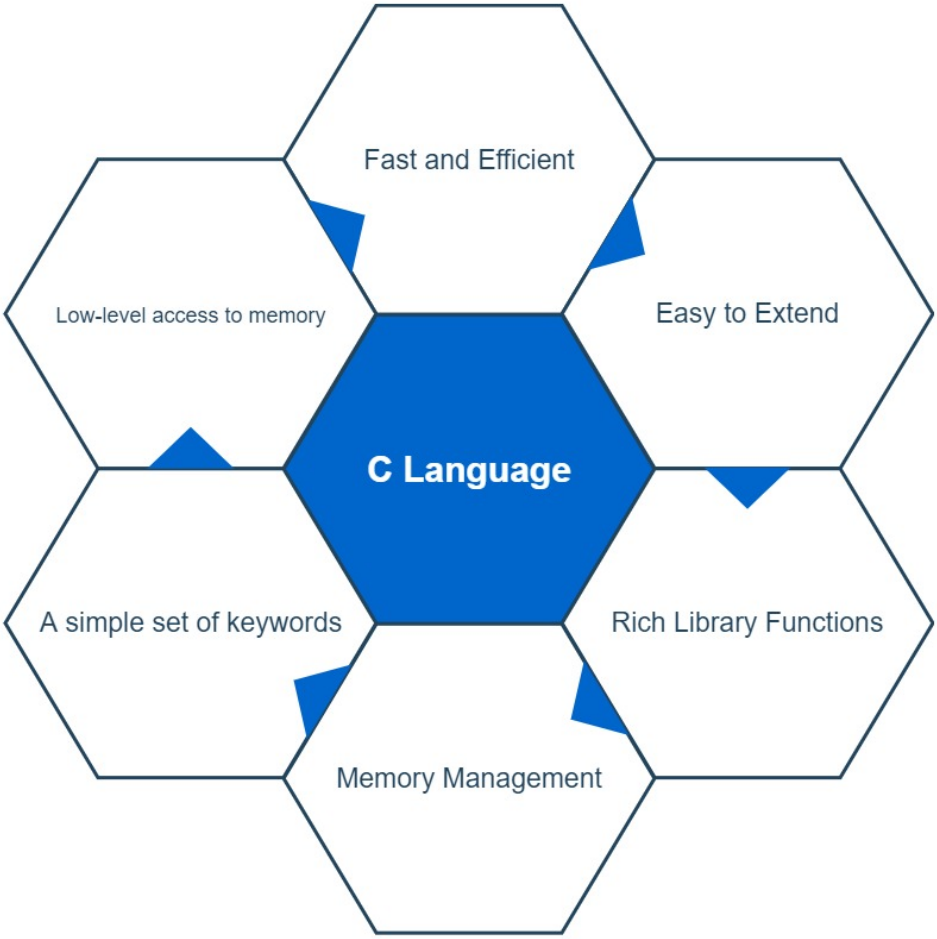
Features Of C Language

There are **different types** of features are available in the C language.

All the features are not possible to mention in one article.

Although, some of the **key features** are mentioned here:

- Fast and Efficient
- Easy to Extend
- Procedural Language
- Simple and clean style
- Middle-Level Language
- Low-level access to memory
- Libraries with rich Functions
- Rich set of built-in Operators
- A simple set of keywords
- Support memory management



These features make C language **suitable** for system programs like an operating system or compiler development.

Later programming languages have borrowed **syntaxes** and features directly or indirectly from C language.

Java, PHP, JavaScript, and many other programming languages are mainly based on C language.

Note: C++ is almost a superset of C (very few programs can be compiled with C, but not with C++).

Data Types

Each variable contains a specific data type.

Data types are used to define the **data storage format**.

Each data type requires different amounts of memory space and has some specific features.

There are mainly **4 data types** that are mostly used in c programming.

Those are described here.

- **int**: It is used to store an integer type value (numbers).
- **char**: It stores a single character (alphabets).
- **float**: It is used to store decimal numbers (floating-point value) with single precision.
- **double**: It is also used to store decimal numbers (floating-point value) with double precision.

An **int** is **signed** by **default**.

It means it can represent both **positive** and **negative** values.

On the other hand, an **unsigned int** can never be **negative**.

All data types are listed here.

Data Type	Memory (Bytes)	Range	Format specifier
short int	2	-32768 to 32767	%hd
unsigned short int	2	0 to 65535	%hu
unsigned int	4	0 to 4294967295	%u
int	4	-2147483648 to 2147483647	%d
long int	8	-2147483648 to 2147483647	%ld
unsigned long int	8	0 to 4294967295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18446744073709551615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	16		%Lf

You can also use the **sizeof()** operator to check the **size** of any variable.

Variables

A variable is a simple **word** or **letter** that allocates some **space** in memory.

Basically, a variable used to **store** some different types of **data**.

Different types of variables require different amounts of memory and have some specific set of operations that can be applied to them.

```
/*variable declaration*/
int a;
char b;
float c;
```

Rules For Defining Variables

1. A variable can have any **alphabet**, **digit**, and **underscore**.
2. A variable name must start only with the alphabet, and underscore. It can't start with a **digit**.
3. **No space** is allowed within the variable name.
4. A variable name can not be any **reserved word** or **keyword**. (e.g. int, void, etc.)

Arrays

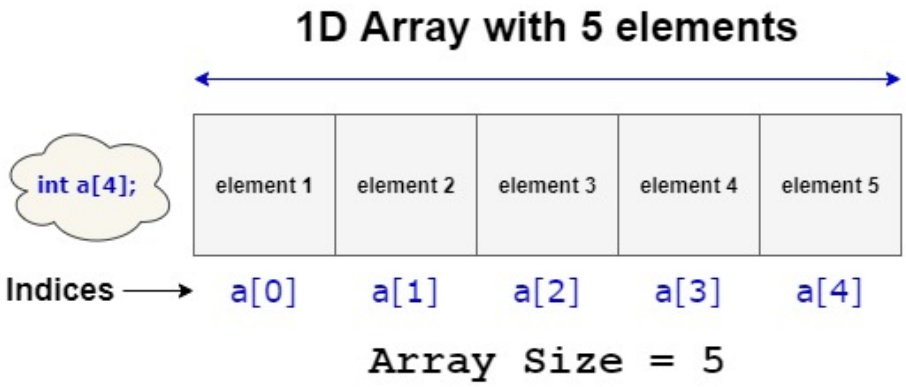
An array is a **data structure** that contains the same types of data items.

A variable can carry only one data item at a time.

If you want to store **multiple data items** in a data type, you need to use an array.

You can not initialize an array with more elements than the specified size.

The specified size is declared to the left of the variable between the**third brackets**.



A **one-dimensional** array is like a **row list**.

On the other hand, a **two-dimensional** (2D) array is like a **table**.

Arrays consist of contiguous memory locations.

Array Declaration

1. Array declaration by specifying the size

```
int a[5];
```

2. Array declaration by initializing the elements

```
int a[] = { 10, 20, 30, 40 };
```

3. Array declaration by specifying the size and initializing the elements

```
int arr[5] = { 10, 20, 30, 40 };
```

Note: You can use **While** or **For** loops to add values in the variables.

Pointers

A pointer is a variable that stores the **address** of another variable.

For example, an integer variable stores an integer **value**, however an integer pointer stores the **address** of an integer variable.

We use the **unary** operator **&** (ampersand) that returns the **address** of a variable.

```
#include <stdio.h>
int main()
{
int x;
printf("%p", &x);
return 0;
}
```

Here, **&x** print the address of variable **x**.

Keywords

Keywords are specific **reserved words** in C which attached with a specific feature.

The list of keywords includes almost all the words that can help us to use the functionality of the C language.

C does not contain very large number of **keywords**.

However, there are **32 keywords** are available in **C98 language**.

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

C99 reserved **five** more keywords.

_Bool	_Imaginary	restrict	_Complex	inline

C11 reserved **seven** more keywords.

_Alignas	_Atomic	_Noreturn	_Thread_local	_Alignof
_Generic	_Static_assert			

Most of the recently reserved words begin with an **underscore** followed by a **capital letter**.

Because identifiers of that form were previously reserved by the C standard for use only by implementations.

Operators

C supports a rich set of operators, which are different types of **symbols**.

Each operator performs a specific operation with a variable.

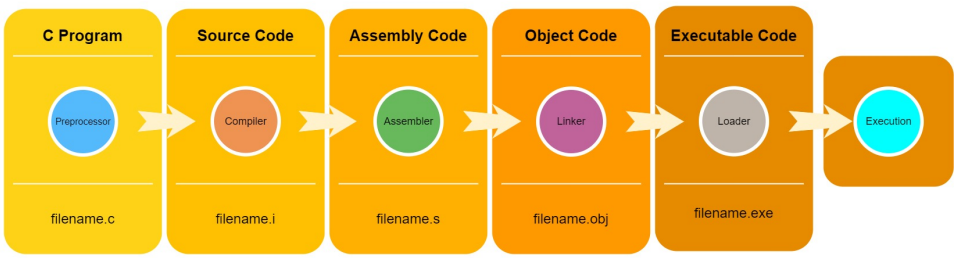
All operators are listed in the following table.

Operator Name	**Operator Symbol**
Arithmetic	+, -, *, /, %
assignment	=
augmented assignment	+=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=
bitwise logic	~, &, |, ^
bitwise shifts	<<, >>
boolean logic	!, &&, ||
conditional evaluation	? :
equality testing	==, !=
calling functions	( )
increment and decrement	++, —
member selection	., ->
object size	sizeof
order relations	<, <=, >, >=
reference and dereference	&, *, [ ]
sequencing	,
subexpression grouping	( )
type conversion	(typename)

These operators tell the compiler to perform specific**mathematical** or **logical** operations.

Memory Management

The most important function of a programming language is to provide facilities for managing memory and objects that are stored in memory.



C language provides **3 unique ways** to **allocate memory** for objects.

Static Memory Allocation

This is an allocation technique that allocates a fixed amount of memory during **compile time**.

Dynamic Memory Allocation

This is also an allocation technique that manages system memory at **runtime**.

Automatic Memory Allocation

When you declare an **automatic variable** (such as a function argument or a local variable), then it happens.

Libraries

Library functions are **inbuilt** functions in C language that are grouped together in common files. This file is called the C standard library.

Each library provides specific functions to perform specific operations.

We can use these library functions to get the pre-defined output instead of writing your own huge complex code to get those outputs.

All C standard library functions are declared in header files which are saved as **filename.h**.

We are including the library in the **header files** in our C program.

```
#include<filename.h>
```

The command allow to use of the functions that are declared in the header files.

Basic Structure Of C Program

A set of **rules** is defined for the C programs that are called **protocols**.

The protocols help us to **design** the basic structure of a program.

Here, I mentioned the **basic structure** of a C program.

- Documentation section
- Link section
- Definition section
- Global declaration section
- Main function section
- Sub-program section

All C programmers must follow the **protocols** when writing any program.

Let's discuss all the **basic structure** sections of a C program.

Documentation Section

The documentation section is a part of the program where the programmers provide the **details** about the program.

In this section programmers usually give the **name** of the program and the **details** related to the program.

This code gives an **overview** of the program.

```
//program name
/*This is a
C Program*/
```

Link Section

This section is used to **declare** all the **header files** that will be used in the program.

It tells the compiler to **link** the header files to the system library.

```
#include<stdio.h>
```

Definition Section

In this section, we can **define** different types of **constants**.

The keyword **define** is used to define a constant value in this part.

```
#define PI=3.14
```

Global Declaration Section

All the **global variables** are declared in this section.

User-defined functions are also declared in this section of the code.

```
int a,b,c;
```

Main Function Section

Every C-programs **must** have the main function.

The main function contains **2 parts**.

1. Declaration Part: All the **variables** are declared in this part.

2. Execution Part: This part starts with the **curly brackets** and ends with the curly close bracket.

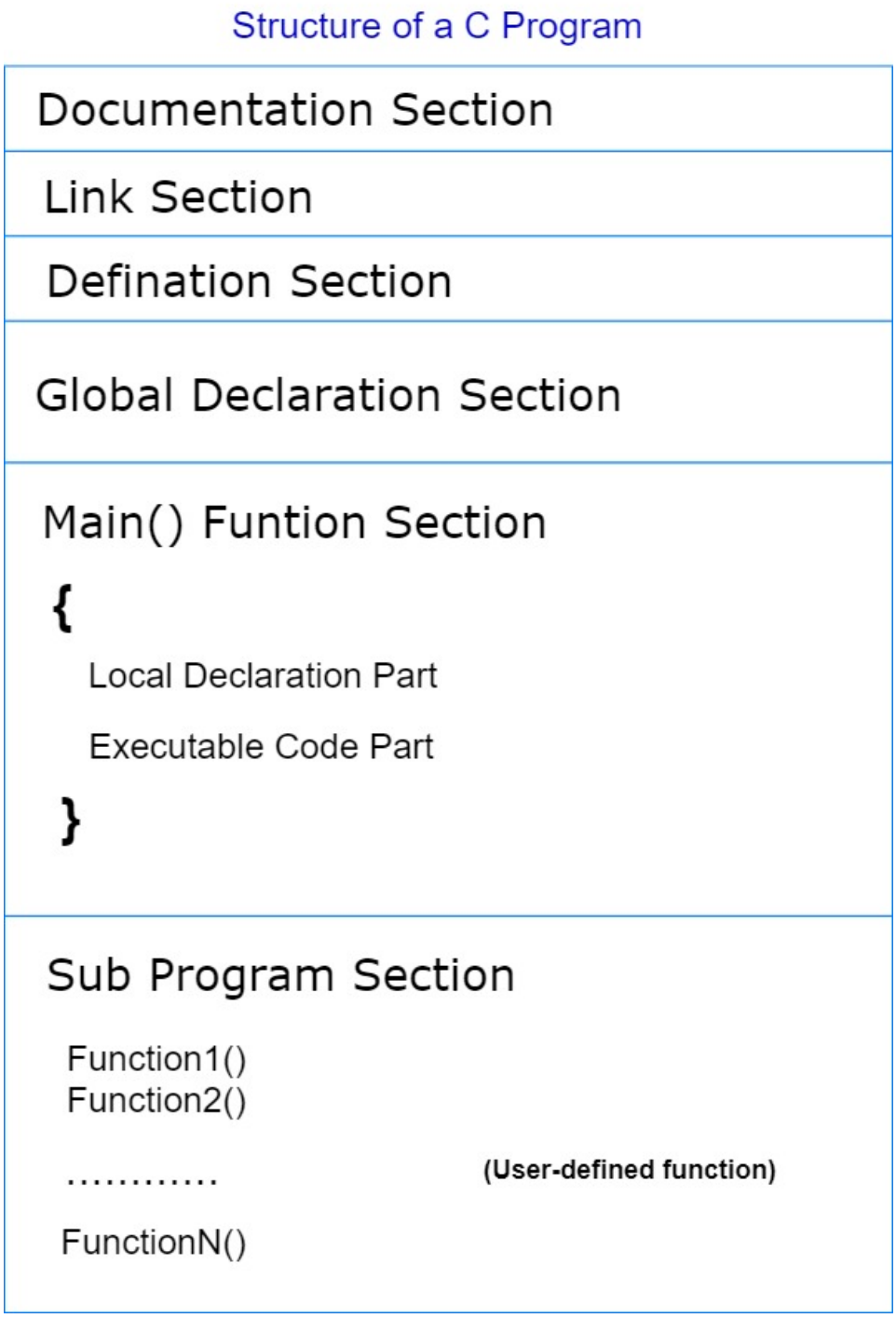
Both the **declaration** and the **execution** part are writing **inside** the curly braces.

```
int main()
{
int a=5;
printf(" %d", a);
return 0;
}
```

Sub-program Section

All user-defined functions are defined in this section.

```
int add(int a, int b)
{
return a+b;
}
```



Hello World C Program

This is the source code of a basic "**Hello World**" Program.

```
#include<stdio.h>
int main()
{
/*First basic C Program*/
printf("Hello World.");
getch();
return 0;
}
```

After **compiling** the source code the output will be the following:

Output:

Hello World.

Explanation of “Hello World” C Program

Here, I explained each line of the “**Hello World**” C program.

```
#include <stdio.h>
```

This is a preprocessor command that includes the input **header file** from the C library before compiling a program.

```
int main()
```

This is the **main function** of executing any C program begins.

```
{
```

It represents the beginning of the main program.

```
/*First basic C Program*/
```

If any words exist inside the command /* and */ in any C program that won’t be considered for compilation and execution. This is also called a **comment line**.

```
printf(“Hello World.“);
```

The **printf** command **displays** the words in the quote on the screen.

```
getch();
```

This function is used to hold the output screen and wait until the user gives any type of input. So that we are able to see the output on the screen.

```
return 0;
```

Here, the return is a keyword that is used to return some value from a function.

The main function returns an **integer** value, therefore here we are returning **0**.

It means our program has been **run** successfully and we terminate our **main function** with this return statement.

```
}
```

It represents the ending of the main program.

Create a C Program

Are you want to **create** and **execute** a C programs yourself?

Then you need to follow the instructions:

1. At first, you need to **install** a C supported **IDE** (Integrated Development Environment) on your computer.
2. Once the IDE is installed on your computer, you can **open** and **create** a C program.

If you don't want to install the **IDE** on your computer, you can use an **online compiler** or **IDE**.

The good thing about the **online compiler** is it can compile **C**, **C++**, **C#**, **Java**, and many other programming languages.

We also provide some **links** to the online and offline **IDE** in this article that can help you to create and execute your C program easily.

Best IDE For C

You can create and edit C programs with any code editor or even a general editor.

Yet, it is very important to choose the **best IDE** for **beginners**.

If the IDE is integrated with the C compiler, the process of **creating** and **compiling** the C program will be easier.

Anyway, we collect some **best IDE** for c program that can help you to write and execute any c program easily.

Here are some collection,

Run C Program Online

- Onlinegdb IDE
- Tutorialspoint IDE
- Rextester IDE

Run C Program On Android Phone

- TruboCdroid
- Cxxdroid
- TurboCPlus
- CppDroid

Run C Program On Windows

- Turbo C++
- Dev C++

- Code::Blocks IDE

Run C Program In Mac OS

- Turbo C++
- Code::Blocks IDE

Run C Program In Linux

- Code::Blocks IDE

Choose the best IDE that makes you **comfortable** to create and edit the C program.

Thus, your **programming skills** will increase and you will be able to **create** any program within a few minutes.

Advantages Of C Language

- It is one of the most useful programming languages when the system requires **quick** and **direct** access to the **hardware**.
- C is the most commonly used system with **limited resources** (such as memory).
- Where **performance** is the most important attribute, C is the best choice for programmers.

Disadvantages Of C Language

- C does not support **OOP** (Object-oriented programming) concepts, that's why C++ is developed.
- There is **no runtime checking** ability in the C language. It only does compile-time checking.
- It does not support the concept of the **namespace**. We cannot declare two variables of the same name without namespace.
- It does not have the concept of **constructor** and **destructor**.

Uses Of C Language

There are **different types** of uses of C language in **programming**.

Some uses are the following:

- C mainly used to **develop** system software, operating systems, BIOS, Embedded Systems, Real-time systems.
- To develop application software like **databases** (MySQL) and **3D software** (Autodesk Maya).
- Used to create **graphical** related applications like computers and mobile games.
- To evaluate any types of **logical** and **mathematical** equations using c language.
- UNIX **kernel** is completely made in C Language.
- The language is used to design different language **compilers**.

Conclusion

The C language doesn't seem to have an **expiration date**.

It has a closeness to the **hardware**, great **portability**, and deterministic usage of **resources**.

For these features, it is the **ideal** programming language for **low-level** development of things like operating system **kernels** and embedded software.

Its good performance, efficiency, and versatility make it an **excellent choice** to develop highly **complex** data manipulation software like MySQL, 3D animation, and more.

C is still unsurpassed where **performance** is the main priority.

I hope now you know all the **basic information** about the C language.