INTRODUCTION

C is a structured, mid-level, general purpose programming language created by **Dennis Ritchie** in the year **1972-73** . C is called structured programming language because a program in c language can be divided into small logical functional modules or structures with the help of function procedure.

C programs are portable i.e., programs written in one type of computer system can run on another type of computer system. C Language was developed in 1970’s at Bell labs

C contains certain additional features that allows it to be used at a lower level, acting as bridge between machine language and the high-level languages.

It was developed for programming in os such as Unix. The standards for C language was introduced known as ANSI C

**ANSI (American National standards Institute).**

What language does computer understand?

The only language that the computer can process or execute is called machine language. It consists of only 0s and 1s in binary, that a computer can understand.

**Programming language**

* Low Level-Machine language and assembly language
* High Level

**Machine language**

Computers can understand only digital signal, which are binary digits i.e., 0 and 1. Writing program in machine level language is a difficult task. Also, machine language programs are not portable.

**Assembly level language**

Assembly language is a low-level programming language for a computer or other programmable device specific to a particular computer architecture in contrast to most high-level programming languages, which are generally portable across multiple systems. Instructions used were MOV, ADD, SUB…Since the computer can understand only machine level language, hence assembly level language must be translated into machine level language. Here Translator used is assembler.

**Assembler-** Converting the code of low-level language (assembly language) into machine language.

**High Level Language**

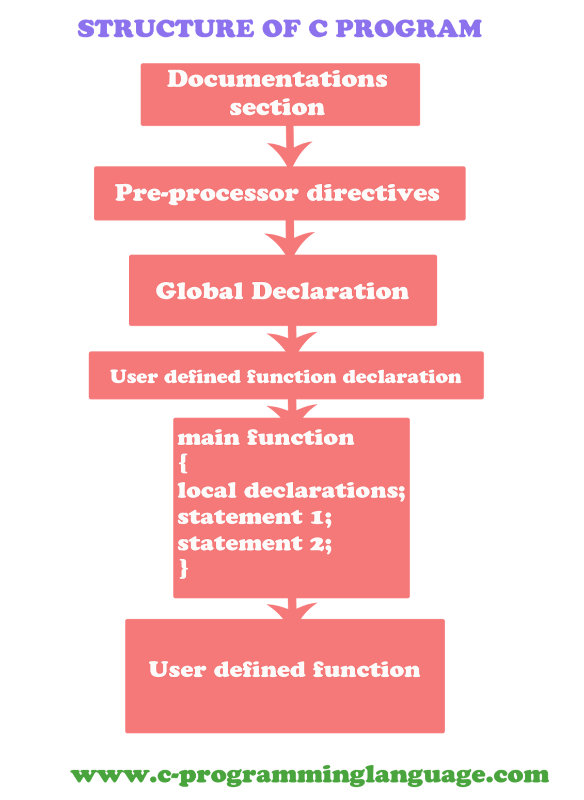
The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages. When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem. A compiler is required to translate a high-level language into a low-level language.

|  |  |
| --- | --- |
| Compiler | Interpreter |
| The compiler scans the whole program in one go. | Translates the program one statement at a time. |
| Execution of the program takes place only after the whole program is compiled. | Execution of the program happens after every line is checked or evaluated. |
| It does not require source code for later execution. | It requires source code for later execution. |
| C, C++, C# etc. | Python, Ruby, Perl, SNOBOL, MATLAB etc. |

**Structure of C program**

/\*Documentation Section**:**

Program Name: Program to find the area of circle.

\*/

#include<stdio.h> //link section

#define PI 3.14 //definition section

float area; //global declaration section

void message(); //Function Declaration Section

void main()

{

float r; //declaration part

printf("Enter the radius of the circle\n"); //executable part

scanf("%f",&r);

area=PI\*r\*r;

printf("Area of the circle=%f \n",area);

message();

}

void message() // Sub Program Section

{

printf("This Sub Function \n");

printf("we can take more Sub Function \n");

}

1. Documentation - It consists of the description of the programs.
2. Pre-processor Section - All the header files of the program will be declared in

the [pre-processor](https://www.geeksforgeeks.org/cc-preprocessors/) section of the program.

1. Definition- - There are multiple steps which are involved in the writing and

execution of the program.

1. Global Declaration - The global declaration section contains global

variables, function declaration, and static variables

1. Main() Function - The main() function of the program is written in this section.

Operations like declaration and execution are performed inside the

curly braces of the main program.

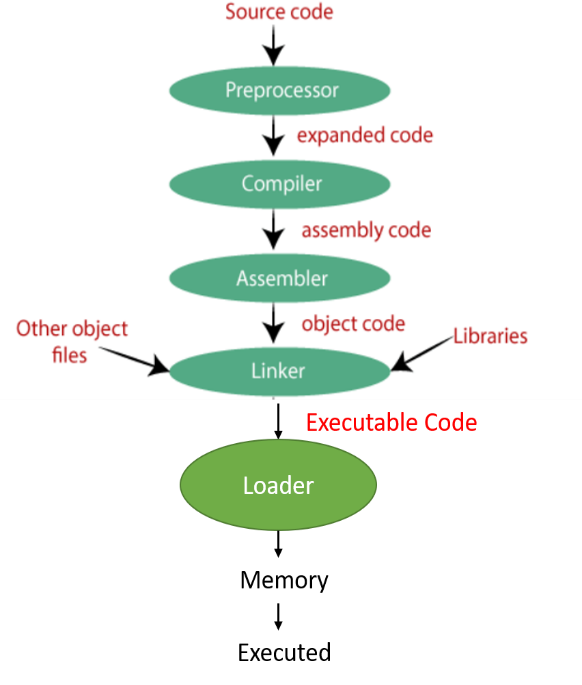
1. Sub Programs - User-defined functions are called in this section of the program. The

control of the program is shifted to the called function whenever they

are called from the main or outside the main() function.

**Question-1) Print hello world. 2) Sum of two/three Number. 3) Do basic mathematical calculation b/w two numbers. 4) Print student details as NAME COURSE ROLL NUMBER PERCENTAGE.**

Compilation phase of C program

* **SOURCE CODE-** Source code is generated by human

or programmer. Source code is high

level code. Source code is written in

plain text by using some high-level

programming language.

* **Pre-processor -** The source code is the code which is

written in a text editor and the source

code file is given an extension ".c".

This source code is first passed to the

pre-processor, and then the pre-

processor expands this code. After

expanding the code, the expanded code

is passed to the compiler.

**Compiling -** The next step is to compile filename.i

and produce an; intermediate compiled

output file **filename.s**. This file is in

assembly level instructions.

**Assembly -** In this phase the filename.s is taken as

input and turned into **filename.o** by

assembler. This file contain machine

level instructions.

**Linker -** This is the final phase in which all the

linking of function calls with their

definitions are done. Linker knows

where all these functions are

implemented.

**Loader -** A loader is a program used by an

operating system to load programs from

a secondary to main memory so as to be

executed.

**Program Design Methodologies**

* Top-down
* bottom-up design approaches
* Modular approach

**Top-down**

Top-down design method starts with top level components to lowest level component.

In Top-down Model, the focus is on breaking the bigger problem into smaller one and then repeat the process with each problem.

It is followed by structural programming like c , Fortran etc.

**bottom-up design approaches**

It starts from the lowest level to the highest-level component.

It first designs the basic components and from these basic components higher level components are designed.

In Bottom-Up Model, the focus is on identifying and resolving smallest problems and then integrating them together to solve the bigger problem.

Bottom-Up Model is mainly used by object-oriented programming languages like Java, C++ etc.

**Modular approach**

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules.

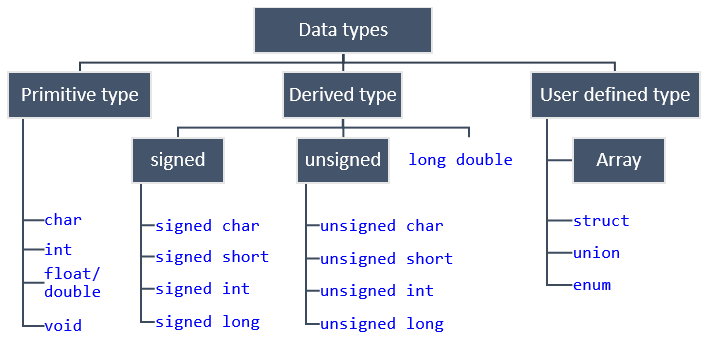
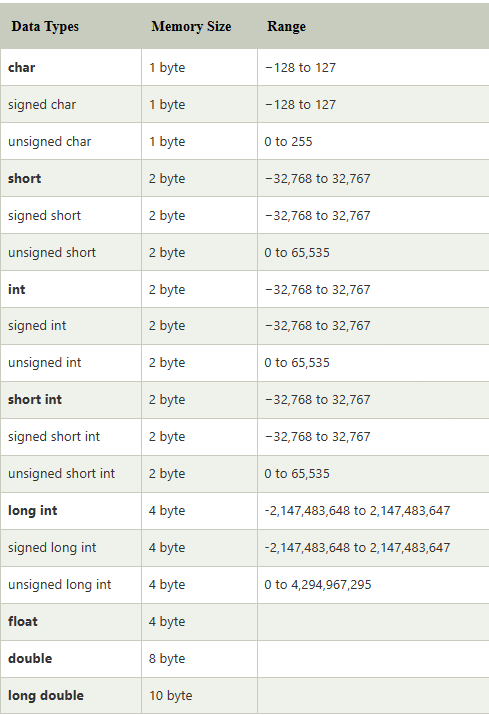
Each module is a separate part of a program.

The process of subdividing a computer program into separate sub programs.

**Data type**

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. It specifies the type of data that the variable can store like integer, character, floating, double, etc. The data type is a collection of data with values having fixed values, meaning as well as its characteristics.

|  |  |
| --- | --- |
| **Types** | **Description** |
| Primitive Data Types | Arithmetic types can be further classified into integer and floating data types. |
| Void Types | he data type has no value or operator and it does not provide a result to its caller. |
| User Defined Datatypes | it is mainly used to assign names to integral constants, which make a program easy to read and maintain. |
| Derived types | The data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. |

****

**Question 1) Use sizeof() Function. To check the size of data type in your system.**

**(example printf(“Integer=%d”,sizeof(int));**

# **Format specifiers in C**

The format specifier is used during input and output. It is a way to tell the compiler what type of data is in a variable during taking input using scanf() or printing using printf().

# **Keywords in C**

Keywords are predefined or reserved words that have special meanings to the compiler. These are part of the syntax and cannot be used as identifiers in the program.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

**There are 32 keywords in C.**

**Variable in C**

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

Declaration of variable= type **variable\_name=value;**

Constant in C

A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming" etc.

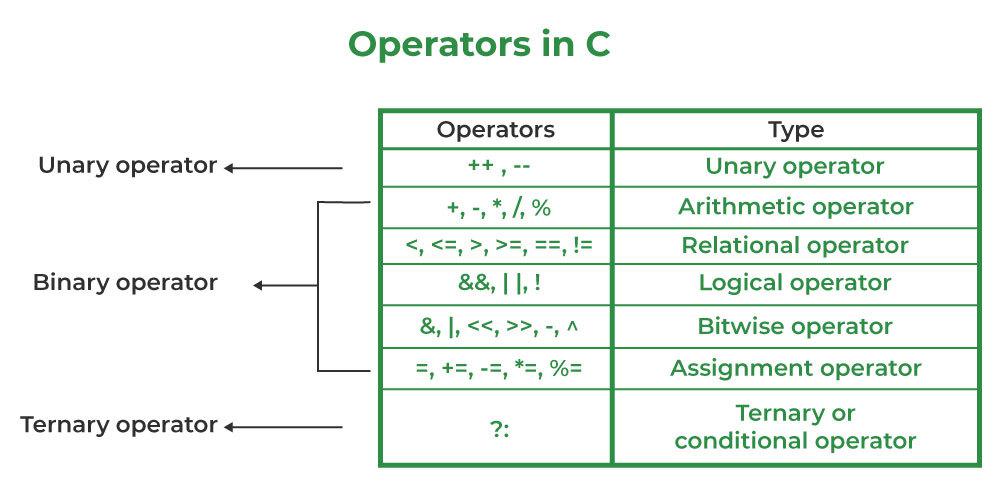
**Identifier in C**

An identifier is a collection of alphanumeric characters that begins either with an alphabetical character or an underscore, which are used to represent various programming elements such as variables, functions, arrays, structures, unions, labels, etc.

### **Modifier in C**

Modifiers are keywords in c which changes the meaning of basic data type in c. It specifies the amount of memory space to be allocated for a variable. Modifiers are prefixed with basic data types to modify the memory allocated for a variable. There are five data type modifiers in C Programming Language:

* long
* short
* signed
* unsigned
* long long

**Operators in c**

**Operators** are the foundation of any programming language. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands.

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Other Operators

**Switch case**

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.