**Programming in C**

**( History , features ,Basic structure )**

***Prepared By***
Alok Haldar

Assistant professor
Department of Computer Science & BCA
Kharagpur College

**List of Books for Programming in C (BCA-1102)**

**1. Programming in C by *T. Jeyapoovan***

**2. Let us C by *Yashvant Kanetkar***

**3. Programming in C by *E.Balaguruswamy***

**4. The Complete Refence By *Herbert Schildt***

**5. Programming in C by *Byron Gottfried***

**6. C programming language by *Brian Kernighan and Dennis Ritchie***

**Programming in C**

It can be defined by the following ways:

1. Procedure-oriented programming language
2. Structured programming language
3. Mid-level programming language

## C as a procedural language

A procedure is known as a function, method, routine, subroutine, etc. A procedural language specifies a series of steps for the program to solve the problem.

A procedural language breaks the program into functions, data structures, etc.

C is a procedural language. In C, variables and function prototypes must be declared before being used.

## C as a structured programming language

A structured programming language is a subset of the procedural language. Structure means to break a program into parts or blocks so that it may be easy to understand.

In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

## C as a mid-level programming language

C is considered as a middle-level language because it supports the feature of both low-level and high-level languages. C language program is converted into assembly code, it supports pointer arithmetic (low-level), but it is machine independent (a feature of high-level).

A Low-level language is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand.

A High-Level language is not specific to one machine, i.e., machine independent. It is easy to understand.

# History of C Language



History of C language is interesting to know. Here we are going to discuss a brief history of the c language.

**C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

**Dennis Ritchie** is known as the **founder of the c language**.

Initially, C language was developed to be used in **UNIX operating system**.

# Features of C Language

 C is the widely used language. It provides many features that are given below.

1. Simple
2. Mid-level programming language
3. Structured programming language
4. Fast Speed
5. Pointers
6. Recursion

# How to install C

There are many compilers available for c and c++. You need to download any one. Here, we are going to use Turbo C++. It will work for both C and C++. To install the Turbo C software, you need to follow following steps.

1. Download Turbo C++
2. Create turboc directory inside c drive and extract the tc3.zip inside c:\turboc
3. Double click on install.exe file
4. Click on the tc application file located inside c:\TC\BIN to write the c program

# First C Program

To write the first c program, open the C console and write the following code:

1. #include<stdio.h>
2. int main()
3. {
4. printf("Hello C Language");
5. return 0;
6. }

#include <stdio.h> includes the standard input output library functions. The printf() function is defined in stdio.h .

int main() The main() function is the entry point of every program in c language.

printf() The printf() function is used to print data on the console.

return 0 The return 0 statement, returns execution status to the OS. The 0 value is used for successful execution and 1 for unsuccessful execution.

# Compilation process in c

## What is a compilation?

The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.



The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps,

i.e., Pre-processing, Compiling, Assembling, and Linking.

The preprocessor takes the source code as an input, and it removes all the **comments** from the source code. The preprocessor takes the preprocessor directive and interprets it.

For example, if <stdio.h>, the directive is available in the program, then the preprocessor interprets the directive and replace this directive with the content of the 'stdio.h' file.

Let's understand through an example.

hello.c

1. #include<stdio.h>
2. int main()
3. {
4. printf("Hello C");
5. return 0;
6. }

Now, we will create a flow diagram of the above program:

# printf() and scanf() in C

The printf() and scanf() functions are used for input and output in C language. Both functions are inbuilt library functions, defined in stdio.h (header file).

### printf() function

The printf() function is used for output. It prints the given statement to the console.

The syntax of printf() function is given below:

1. printf("format string",argument\_list);

The format string can be %d (integer), %c (character), %s (string), %f (float) etc.

### scanf() function

The scanf() function is used for input. It reads the input data from the console.

1. scanf("format string",argument\_list);

**Sample Program:**

1. #include<stdio.h>
2. int main(){
3. int number;
4. printf("Enter a number:");
5. scanf("%d",&number);
6. printf("Square of number is:%d",number\*number);
7. return 0;
8. }

The scanf("%d",&number) statement **reads integer number from the console** and **stores the given value in number variable.**

The printf("Square of number is:%d ",number\*number) statement prints the square of number on the console.

### Program to print sum of 2 numbers

Let's see a simple example of input and output in C language that prints addition of 2 numbers.

1. #include<stdio.h>
2. int main()
3. {
4. int x=0,y=0,result=0;
5. printf("Enter first number:");
6. scanf("%d",&x);
7. printf("enter second number:");
8. scanf("%d",&y);
9. result=x+y;
10. printf("sum of 2 numbers:%d",result);
11. return 0;
12. }

# Variables in C

A variable is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.

It is a way to represent memory location through symbol so that it can be easily identified.

Let's see the syntax to declare a variable:

**type variable\_list;**

The example of declaring the variable is given below:

* int a;
* float b;
* char c;

Here, a, b, c are variables. The int, float, char are the data types.

We can also provide values while declaring the variables as given below:

* int a=10,b=20;//declaring 2 variable of integer type
* float f=20.8;
* char c='A';

## Rules for defining variables

* A variable can have alphabets, digits, and underscore.
* A variable name can start with the alphabet, and underscore only. It can't start with a digit.
* No whitespace is allowed within the variable name.
* A variable name must not be any reserved word or keyword, e.g. int, float, etc.

Valid variable names:

1. int a;
2. int \_ab;
3. int a30;

Invalid variable names:

1. int 2;
2. int a b;
3. int long;

## **Types of Variables in C**

There are many types of variables in c:

1. local variable
2. global variable
3. static variable
4. automatic variable
5. external variable

### **Local Variable**

A variable that is declared inside the function or block is called a local variable.

It must be declared at the start of the block.

1. Void function1()
2. {
3. int x=10;//local variable
4. }

You must have to initialize the local variable before it is used.

### Global Variable

A variable that is declared outside the function or block is called a global variable. Any function can change the value of the global variable. It is available to all the functions.

It must be declared at the start of the block.

1. Int value=20;//global variable
2. void function1()
3. {
4. int x=10;//local variable
5. }

### Static Variable

A variable that is declared with the static keyword is called static variable.

It retains its value between multiple function calls.

1. Void function1(){
2. int x=10;//local variable
3. static int y=10;//static variable
4. x=x+1;
5. y=y+1;
6. printf("%d,%d",x,y);
7. }

If you call this function many times, the local variable will print the same value for each function call, e.g, 11,11,11 and so on. But the static variable will print the incremented value in each function call, e.g. 11, 12, 13 and so on.

### Automatic Variable

All variables in C that are declared inside the block, are automatic variables by default. We can explicitly declare an automatic variable using auto keyword.

1. Void main()
2. {
3. int x=10;//local variable (also automatic)
4. auto int y=20;//automatic variable
5. }

### External Variable

We can share a variable in multiple C source files by using an external variable. To declare an external variable, you need to use extern keyword.

**program1.c**

1. #include<stdio.h>
2. extern int global\_valriable=10;//external variable(also global)
3. int main()
4. {
5. void printValue();
6. printValue();
7. return 0;
8. }
9. void printValue()
10. {
11. printf("Global variable:%d", global\_variable);
12. }

# Data Types in C

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.

There are the following data types in C language.

|  |  |
| --- | --- |
| Types | Data Types |
| Basic Data Type | int, float, char, double |
| Derived Data Type | array, pointer, structure, union |
| Enumeration Data Type | enum |
| Void Data Type | void |

## Basic Data Types

The basic data types are integer-based and floating-point based. C language supports both signed and unsigned literals.

The memory size of the basic data types may change according to 32 or 64-bit operating system.

Let's see the basic data types. Its size is given according to 32-bit architecture.

|  |  |  |
| --- | --- | --- |
| Data Types | Memory Size | Range |
| **char** | 1 byte | −128 to 127 |
| signed char | 1 byte | −128 to 127 |
| unsigned char | 1 byte | 0 to 255 |
| **short** | 2 byte | −32,768 to 32,767 |
| signed short | 2 byte | −32,768 to 32,767 |
| unsigned short | 2 byte | 0 to 65,535 |
| **int** | 2 byte | −32,768 to 32,767 |
| signed int | 2 byte | −32,768 to 32,767 |
| unsigned int | 2 byte | 0 to 65,535 |
| **short int** | 2 byte | −32,768 to 32,767 |
| signed short int | 2 byte | −32,768 to 32,767 |
| unsigned short int | 2 byte | 0 to 65,535 |
| **long int** | 4 byte | -2,147,483,648 to 2,147,483,647 |
| signed long int | 4 byte | -2,147,483,648 to 2,147,483,647 |
| unsigned long int | 4 byte | 0 to 4,294,967,295 |
| **float** | 4 byte |  |
| **double** | 8 byte |  |
| **long double** | 10 byte |  |

# Keywords in C

A keyword is a reserved word. You cannot use it as a variable name, constant name, etc. There are only **32 reserved words (keywords)** in the C language.

A list of 32 keywords in the c language is given below:

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

# C Identifiers :

C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc. An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore. If the identifier is not used in the external linkage, then it is called as an internal identifier. If the identifier is used in the external linkage, then it is called as an external identifier.

# C Operators :

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise, etc.

There are following types of operators to perform different types of operations in C language.

* Arithmetic Operators
* Relational Operators
* Shift Operators
* Logical Operators
* Bitwise Operators
* Ternary or Conditional Operators
* Assignment Operator