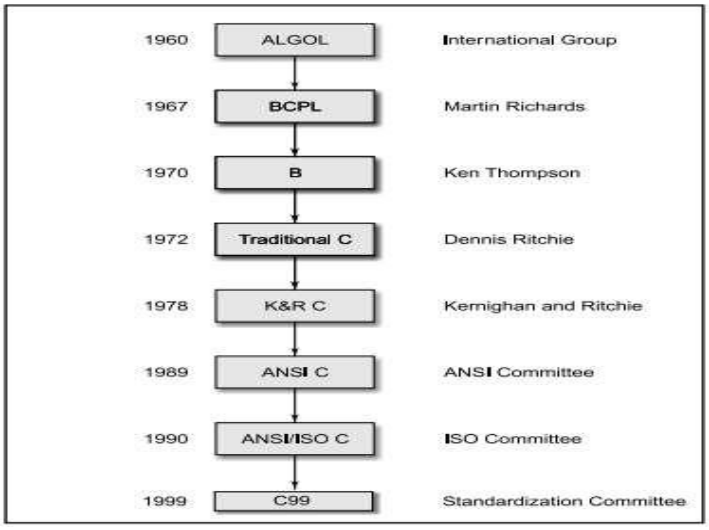
Introduction to C

C programming is a general-purpose, procedural programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. It is machine-independent, structured programming language which is used extensively in various applications.

History of C language

The root of all modern languages is 'ALGOL.' It was first introduced in 1960. 'ALGOL' introduced the concept of structured programming to the developer community. In 1967, a new computer programming language was announced called as 'BCPL' which stands for Basic Combined Programming Language. BCPL was designed and developed by Martin Richards, especially for writing system software. This was the era of programming languages. Just after three years, in 1970 a new programming language called 'B' was introduced by Ken Thompson that contained multiple features of 'BCPL.' This programming language was created using UNIX operating system at AT&T and Bell Laboratories. Both the 'BCPL' and 'B' were system programming languages.



In 1972, a great computer scientist Dennis Ritchie created a new programming language called 'C' at the Bell Laboratories. It was created from 'ALGOL', 'BCPL' and 'B' programming languages. 'C' programming language contains all the features of these languages and many more additional concepts that make it unique from other languages.

American National Standards Institute (ANSI) defined a commercial standard for 'C' language in 1989. Later, it was approved by the International Standards Organization (ISO) in 1990. 'C' programming language is also called as 'ANSI C'. The standardization committee of C added a few features of C++/Java to

enhance the usefulness of the language. The result was the 1999 standard for C. This version is usually referred to as C99. The history and development of C

Where is C used? Key Applications

'C' language is widely used in embedded systems.

It is used for developing system applications.

It is widely used for developing desktop applications.

Most of the applications by Adobe are developed using 'C' programming language.

It is used for developing browsers and their extensions. Google's Chromium is built using 'C' programming language.

It is used to develop databases. MySQL is the most popular database software which is built using 'C'.

It is used in developing an operating system. Operating systems such as Apple's OS X,

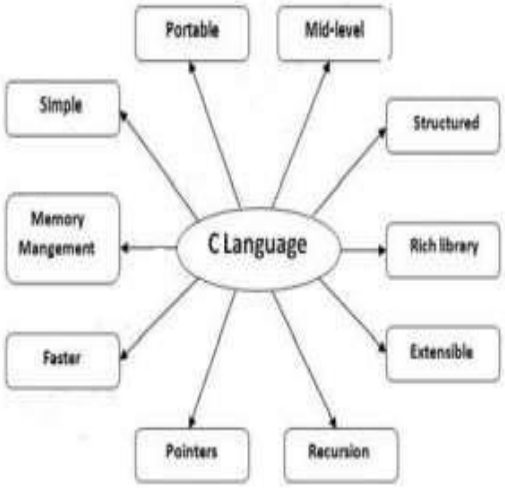
Microsoft's Windows, and Symbian are developed using 'C' language. It is used for

developing desktop as well as mobile phone's operating system.

- It is used for compiler production.
- It is widely used in Internet of Things (IOT) applications.

Features of C

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



Rich Library

It is a robust language whose rich set of built-in functions and operators can be used to write any complex program. C provides a lot of inbuilt functions that make the development fast.

Machine Independent or Portable

C is highly portable. This means that C programs written for one computer can be run on another with little or no modification. The compiler and preprocessor makes it possible to run on different computer or devices.

Mid-level programming language

The C compiler combines the capabilities of an assembly language with the features of a highlevel language and therefore it is well suited for writing both system software and business packages. In fact, many of the C compilers available in the market are written in C.

Structured programming language.

C language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

Recursion

In C, we can call the function within the function. It provides code reusability for every function.

Fast

Programs written in C are efficient and fast. This is due to its variety of data types, functions and powerful operators. It is many times faster than BASIC. For example, a program to increment a variable from 0 to 15000 takes about one second in C while it takes more than 50 seconds in an interpreter BASIC.

Memory Management

It supports the feature of dynamic memory allocation. In C language, we can free the allocated memory at any time by calling the free() function.

Pointer

C provides the feature of pointers. We can directly interact with the memory by using the pointers. We can use pointers for memory, structures, functions, array, etc.

Structure of a C program

C program can be written in this structure only. Writing a C program in any other structure will hence lead to a Compilation Error.

Document Section

 A comment is an explanation or description of the source code of the program. It helps a developer explain logic of the code and improves program readability.

 At run-time, a comment is ignored by the compiler.

There are two types of comments in C:

1) Single line comment

2) Multi Line Comment

Example :

// Single line comment

//Multi Line Comment

//Line 1

//Line 2

(Or)

/* Multi line comment are used in the program

Line 1…

Line 2….

…..,

……

7

*/

Preprocessor Section or Link Section or Header File Section

 The link section provides instructions to the compiler to link functions from the library.

 A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

 stdio.h – Defines core input and output functions

 string.h – Defines string handling functions

 math.h – Defines common mathematical functions

Syntax to include a header file in C:

#include<filename>

Example:

#include<stdio.h>

#include<math.h>

#include<string.h>

Definition Section

 The definition section defines all symbolic constant.

 Symbolic constant is a way of defining a variable constant whose value cannot be changed

Syntax:

#define symbolic constant value

Example

#define MAX 100

#define PI 3.14

Global Declaration Section:

 The variables that are used in more than one function throughout the program are called global variables.

 Should be declared outside of all the functions i.e., before main().

Main Method Declaration:

The next part of a C program is to declare the main() function. Every „C" program must have one main() function where the program execution begins.

Syntax to Declare main method

int main() or void main()

{}

It contains the following two parts

1. Declaration Part

2. Executable part

Declaration Part

It refers to the variables that are to be used in the function. The variables are to be declared before any operation in the function and these are called local variables

Syntax

Data type variable name;

Example:

Void main()

9

{

int a;

}

Execution Part:

It contains at least one valid C Statement. The Execution of a program begins with opening brace "{„ and ends with closing brace "}" .The closing brace of the main function is the logical end of the program.

Sub Program section

 Sub programs are basically functions are written by the user (user defined functions).

 They may be written before or after a main () function and called within main () function.

 This is optional to the programmer.

Constraints while writing a C program

• All statements in „C" program should be written in lower case letters. Uppercase letters are only used for symbolic constants

• Blank space may be inserted between the words. Should not be used while declaring a variable, keyword, constant and function

• The program statements can be written anywhere between the two braces following the

declaration part.

• All the statements should end with a semicolon (;)

Example to illustrate structure of C program

// Program to illustrate structure of C program // Document section //Single line comment

#include<stdio.h> // Link Section

#define PI 3.14; // Definition Section

```
float area(float r); // Global Declaration section
int main() //Main Function
10
{
float r; // Declaration Part
printf(" Enter the radius:n"); // Executable Part
scanf("%f",&r);s
printf("the area is: %f",area(r));
return 0;
}
float area(float r)
{
return pi * r * r; //sub program or User defined function
}
```

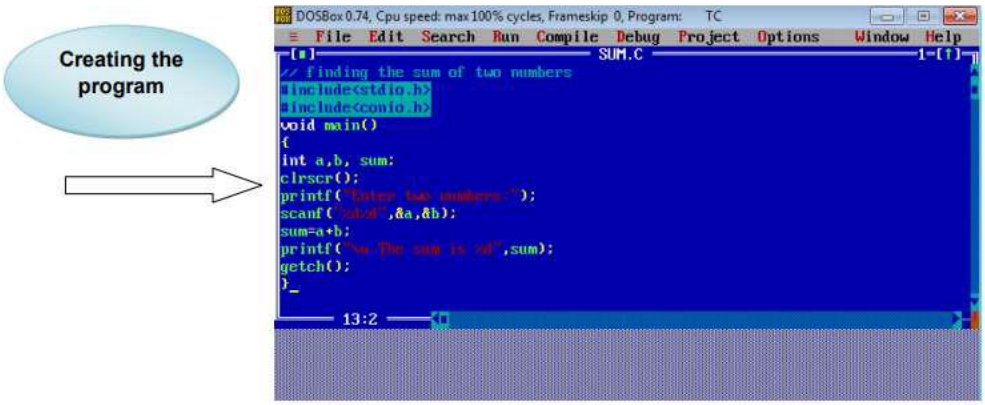
Compilation and Execution of C program

1. Creating the program
2. Compiling the Program
3. Linking the Program with system library
4. Executing the program

Creating the program:

 Type the program and edit it in standard „C" editor and save the program with .c as an extension.

 This is the source program.



Compiling (Alt + F9) the Program:

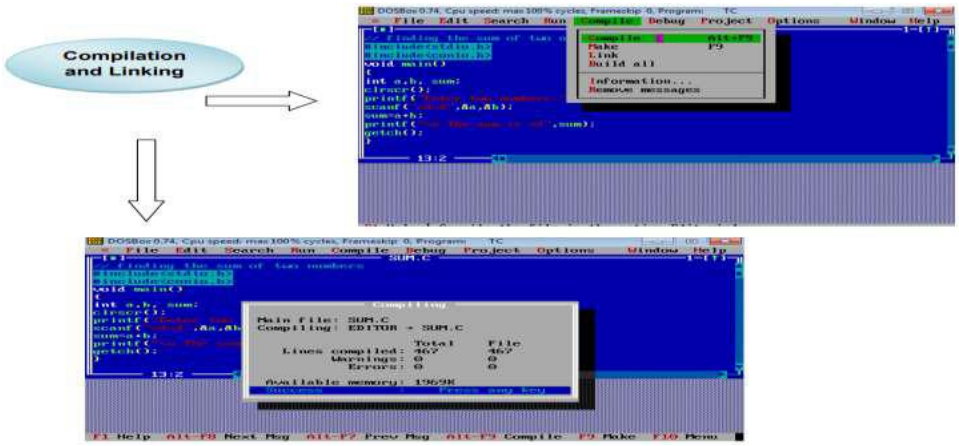
• This is the process of converting the high level language program to Machine level

• Language (Equivalent machine instruction) -> Compiler does it!

• Errors will be reported if there is any, after the compilation

• Otherwise the program will be converted into an object file (.obj file) as a result of the compilation

• After error correction the program has to be compiled again

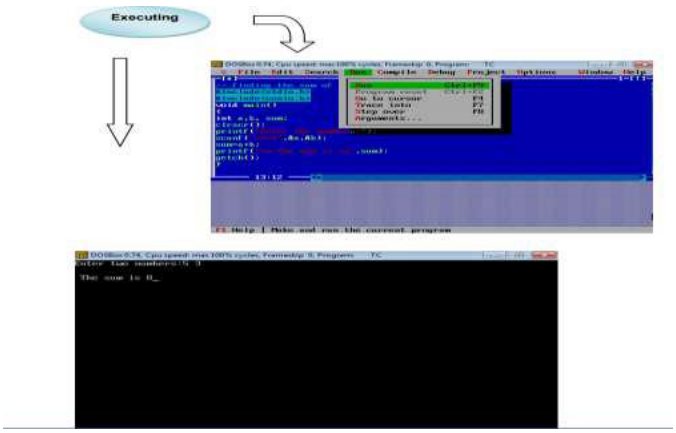


Linking the program with system Library

 Before executing a c program, it has to be linked with the included header files and other system libraries -> Done by the Linker

Executing the Program:

 This is the process of running (Ctrl + F9) and testing the program with sample data. If there are any run time errors, then they will be reported.



Introduction to Compilation and Execution

Compilation and execution are fundamental processes in computer programming that allow you to turn high-level source code into machine-executable code and run it on a computer. These processes are crucial in the software development life cycle and involve multiple steps.

Compilation:

Compilation is the process of translating human-readable high-level source code written in programming languages like C, C++, Java, or Python into machinereadable low-level code, often in the form of binary machine code or bytecode. This translation is done by a special software tool called a compiler or interpreter, depending on the programming language.

The preprocessed source code is translated into an intermediate representation or machine code specific to the target platform. This intermediate representation is often called object code or bytecode.

Preprocessing:

In many programming languages, the source code undergoes preprocessing. This stage typically involves removing comments, expanding macros, and including header files or dependencies.

Linking (for compiled languages)

If the code is written in a compiled language, such as C or C++, the compiler generates separate object files for each source file. The linker combines these object files and resolves dependencies to create an executable binary.

Executable Generation (for compiled languages):

In the case of compiled languages, the final executable binary is produced. This binary contains the machine code that can be executed by the computer's processor.

Execution

Execution is the process of running the compiled or interpreted code on a computer. When you execute a program, the operating system loads the program into memory, and the CPU (Central Processing Unit) processes the instructions in the program sequentially.

Here are the steps involved in program execution:

Loading:

The operating system loads the compiled executable or the interpreter loads the source code. This process involves reserving memory for the program's data and code.

Initialization

If necessary, the program performs any required initialization, such as setting up data structures, opening files, or establishing network connections.

Execution

The CPU fetches instructions from memory and executes them one by one. These instructions may involve arithmetic operations, control flow, I/O operations, and interactions with external resources.

Termination

The program may run until it completes its task, encounters an error, or is manually terminated by the user. When the program terminates, it releases any allocated resources and returns control to the operating system.

Output

During execution, the program may produce output, which can be displayed to the user or saved to files.

Cleanup

After execution, the operating system or interpreter may perform cleanup tasks, such as releasing memory and closing open files.

Compilation and execution are essential stages in the software development process. Compilation translates high-level code into machinereadable form, while execution runs the program on a computer, producing the desired results or performing specific tasks. Understanding these processes is crucial for software developers and computer scientists to create and run software applications effectively.