Unit – 1

INTRODUCTION OF C

HISTORY OF C:

The history of C-language is interesting to know. The C-language is a general-purpose and procedural-oriented programming language. It is a structured and machine-independent programming language. It was developed by Dennis Ritchie in 1972 at the AT&T Bell Laboratories. It was developed along with the UNIX operating system, and is strongly linked with UNIX operating system. History of C language revolves around development as a system implementation language to write an operating system. In terms of the history of C language, its main features include low-level memory access as well as high-level memory access (so it is a middle-level programming language), a handy set of keywords, and a neat and clean style, these features make C programming language suitable for system programming. C supports a wide variety of built-in functions, standard libraries and header files. It follows a top-down approach. Many languages have derived syntax directly or indirectly from the C programming language. For example, C++ is closely a superset of the C language. Also, C programming language is very popular for system-level apps.

What is C?

The C programming language is a procedural and general-purpose language that provides low-level access to system memory. A program written in C must be run through a C compiler to convert it into an executable that a computer can run. Many versions of Unix-based operating systems (OSes) are written in C and it has been standardized as part of the Portable Operating System Interface (POSIX).

Today, the C programming language runs on many different hardware platforms and OSes such as Microsoft and Linux.

Pros and cons of C

The C language comes with a set of special characteristics, making it one of the most widely used languages of all time. The following are the main benefits of using C:

- **Structured.** It offers a structured programming approach for breaking down problems into smaller modules or functions that are easy to understand and modify.
- **Portable.** C is machine-independent and C programs can be executed on different machines.

- **Mid-level programming language.** It's a mid-level language that supports the features of both a low-level and a high-level language.
- **Rich library.** It offers numerous built-in library functions that expedite the development process.
- **Dynamic memory allocation.** C supports the dynamic memory allocation feature, which can be used to free the allocated memory at any time by calling the **free()** function.
- **Speed.** It's a compiler-based language, which makes the compilation and execution of code faster. Since only essential and required features are included in C, it saves processing power and improves speed.
- **Pointers.** C uses pointers, which improve performance by enabling direct interaction with the system memory.
- **Recursion.** C enables developers to backtrack by providing code reusability for every function.
- **Extensible.** A C program can be easily extended. If code is already written, new features and functionalities can be added to it with minor alterations.

C also comes with a few shortfalls, even though it's an ideal language for programming beginners due to its simple syntax, algorithms and modular structure. The following are a few disadvantages of using C:

- **OOP features.** C doesn't extend its support for object-oriented programming (OOP) features, which enables the creation of subclasses from parent classes. Unlike Java, Python or C++, multiple inheritances can't be created in C, which makes it difficult to reuse existing code.
- **Namespace feature.** C lacks namespace features, which means the same variable name can't be reused in one scope. Without namespaces, it's impossible to declare two variables with the same name.
- **Run-time checking.** C doesn't display code errors after each line of code; instead, all the errors are presented by the compiler after the program has been written. This can make code checking a challenge, especially for larger programs.

- **Exception handling.** C lacks exception handling, which is the ability to handle exceptions, such as bugs and anomalies that can happen during source code
- **Constructor and destructor.** Since C isn't object oriented, it doesn't offer constructor and destructor features. Constructing or destructing a variable in C must be done manually through a function or by other means.
- **Garbage collection.** C isn't equipped with garbage collection. This important feature automatically reclaims memory from objects that are no longer required by the library or an app.

Where is C used?

The following are some use cases for the C language:

- OSes, such as Unix and all Unix applications;
- databases, including Oracle Database, MySQL, Microsoft SQL Server and PostgreSQL, which are partially written in C;
- language compilers, including the C compiler;
- text editors;
- print spoolers;
- assemblers;
- network drivers;
- modern programs, such as Git and FreeBSD;
- language interpreters; and
- utilities, such as network drivers, mouse drivers and keyboard drivers.

What's the difference between C and C++?

The following highlights the differences between the two languages:

- C is a procedural language that provides no support for objects and classes. C++ is a combination of OOP and procedural programming languages.

- C has 32 keywords and C++ has 63 keywords.
- C supports built-in data types, while C++ supports both built-in and user-defined data types.
- C doesn't have access modifiers, whereas C++ does.
- C uses the **<stdio.h>** header file for input and output operations and C++ uses the **<iostream.h>** header file for input and output operations.
- C can't hide data, while C++ is secure and provides encryption.
- There's no direct exception handling support in C, but C++ supports it.
- C doesn't support function and operator overloading, but C++ does.
- In C, the **main()** function calls are made through other functions used in the code, but C++ doesn't provide that functionality.
- Reference variables aren't supported by C, but C++ supports them.

How does C Programming Language Work?

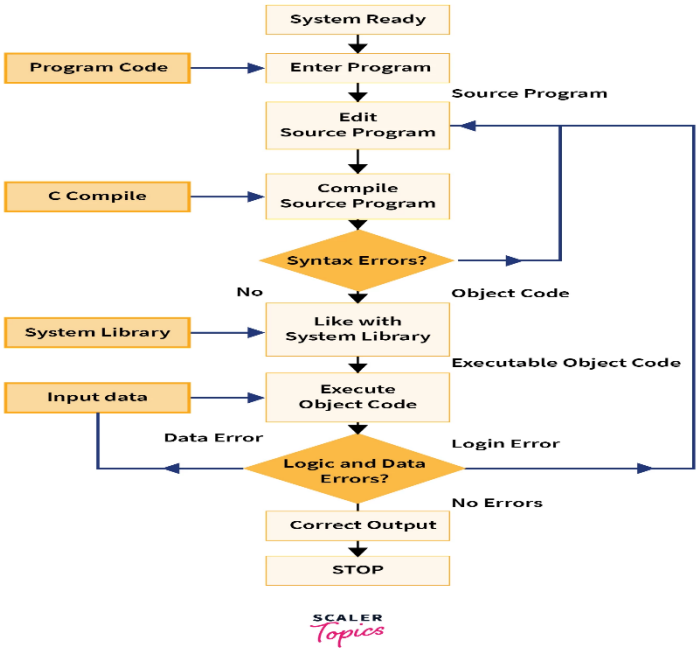
Execution of the C program involves 5 steps. These are:

1. Creating the Program
2. Preprocessing
3. Compiling the Program
4. Linking the Program with functions from the C library
5. Executing the Program

Explaination:

1. **Creating the Program:** Firstly, we need to create a C program. For that, we will open a text editor and write our C program into it. Then save the file with .c extension. For example: hello.c The program written into the file is known as the source code, and it serves the original form of the C program.
2. **Preprocessing**: Preprocessing is the stage where source code is passed for the first time. This stage consists of the following steps:

- Expansion of Macros and Comment Removal
- Expansion of the included files
- Conditional compilation The preprocessed output of hello.c is stored in the file hello.i.



3. **Compiling the Program:** Once our source code is preprocessed in the file hello.i. Now our file is ready for compilation, which after compilation produces an intermediate compiled output file hello.s, which is in assembly level instructions. During the compilation process, the compiler checks for all the compilation errors. If the online C compiler gives no error, then the hello.s is taken as input and turned into hello.o by the assembler in the next phase. This file contains machine-level instructions. At this phase, only existing code is converted into machine language, and the function calls are not resolved. Since the object file is not executable, the process is transferred to the linker, which ultimately produces a .exe file.
4. **Linking the Program with functions from the C library:** This is the final phase, in which all of the function calls are linked to their definitions. Linker knows where all these functions are implemented. The linker performs additional work and adds more code to our programme that is required when it starts and stops. Setting up the environment, for example, requires a code, as does sending command-line inputs. The linker connects the object code of our programme file to the C library functions, resulting in an.exe file, hello.exe, which is an executable file, will be created here.
5. **Executing the Program:** The execution of a program is a very simple task. After giving the command to execute a particular program. The loader will load the executable object code into the RAM and execute the instructions.

PROGRAM DEVELOPMENT STEPS:

Program development life cycle contains 6 phases, which are as follows –

- Problem Definition.
- Problem Analysis.
- Algorithm Development.
- Coding & Documentation.

- Testing & Debugging.
- Maintenance.

- **Problem Definition**

Here, we define the problem statement and decide the boundaries of the problem.

In this phase, we need to understand what is the problem statement, what is our requirement and what is the output of the problem solution. All these are included in the first phase of program development life cycle.

- **Problem Analysis**

Here, we determine the requirements like variables, functions, etc. to solve the problem. It means that we gather the required resources to solve the problem, which are defined in the problem definition phase. Here, we also determine the bounds of the solution.

- **Algorithm Development**

Here, we develop a step-by-step procedure that is used to solve the problem by using the specification given in the previous phase. It is very important phase for the program development. We write the solution in step-by-step statements.

- **Coding & Documentation**

Here, we use a programming language to write or implement the actual programming instructions for the steps defined in the previous phase. We construct the actual program in this phase. We write the program to solve the given problem by using the programming languages like C, C++, Java, etc.

- **Testing & Debugging**

In this phase, we check whether the written code in the previous step is solving the specified problem or not. This means, we try to test the program whether it is solving the problem for various input data values or not. We also test if it is providing the desired output or not.

- **Maintenance**

In this phase, we make the enhancements. Therefore, the solution is used by the end-user. If the user gets any problem or wants any enhancement, then we need to repeat all these phases from the starting, so that the encountered problem is solved or enhancement is added.

STRUCTURE OF C:

Some basic commands are required to write a C programme. But, before we get into the basic C commands, let's have a look at a simple C program.

CODE-

```
#include <stdio.h>

int main() {

   printf("Welcome to the C");

   return 0;

}
```

OUTPUT-

Welcome to the C

Below are a few basic commands of C.

S. No.	C Basic Commands	Name	What it does (Explanation)
1	#include	Preprocessor directive	Used to include header files.
2	<stdio.h>	header file	The stdio.h header defines three variable types, several macros, and a variety of input and output functions.
3	main()	main function	The execution of code begins from main function.
4	{	opening curly brace	It indicates the start of a function.
5	printf()	printing function	Used to display output on the screen.
6	;	semicolon	Marks the end of the statement.
7	return 0;	return 0	This command shows the exit status of a function.
8	}	closing curly brace	It indicates the end of a function.