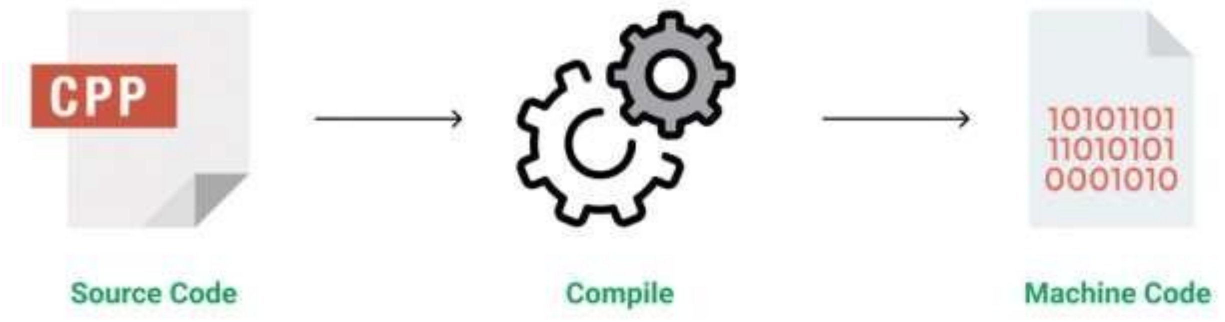
Overview (Introduction to C++ Programming Language)

C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.
C++ is regarded as a middle-level language, as it comprises a combination of both high level and low-level language features.
C++ was developed by ***Bjarne Stroustrup*** starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983.
C++ is a superset of C, and that virtually any legal C program is a legal C++ program.
Note: A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time.



Features of C++

- **Simple:** It is a simple language in the sense that programs can be broken down into logical units and parts, has a rich library support and a variety of data-types.
- **Machine Independent but Platform Dependent:** A C++ executable is not platform-independent (compiled programs on Linux won't run on Windows), however they are machine independent.
- **Mid-level language:** It is a mid-level language as we can do both systems-programming (drivers, kernels, networking etc.) and builds large-scale user applications (Media Players, Photoshop, Game Engines etc.)
- **Rich library support:** Has a rich library support (Both standard ~ built-in data structures, algorithms etc.) as well 3rd party libraries (e.g. Boost libraries) for fast and rapid development.
- **Speed of execution:** C++ programs excel in execution speed. Since, it is a compiled

language, and also hugely procedural. Newer languages have extra in-built default features such as grabage-collection, dynamic typing etc. which slow the execution of the program overall. Since there is no additional processing overhead like this in C++, it is blazing fast.

- **Pointer and direct Memory-Access:** C++ provides pointer support which aids users to directly manipulate storage address. This helps in doing low-level programming (where one might need to have explicit control on the storage of variables).
- **Object-Oriented:** One of the strongest points of the language which sets it apart from C. Object-Oriented support helps C++ to make maintainable and extensible programs. i.e. Large-scale applications can be built. Procedural code becomes difficult to maintain as code-size grows.
- **Compiled Language:** C++ is a compiled language, contributing to its speed.

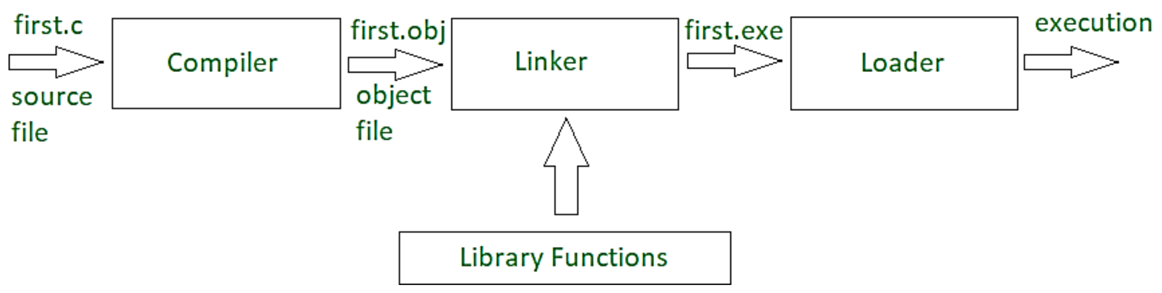
Applications of C++:

C++ finds varied usage in applications such as:

- Operating Systems & Systems Programming. e.g. Linux-based OS (Ubuntu etc.)
- Browsers (Chrome & Firefox)
- Graphics & Game engines (Photoshop, Blender, Unreal-Engine)
- Database Engines (MySQL, MongoDB, Redis etc.)
- Cloud/Distributed Systems

The Compilation Process

A program goes from text files (or source files) to processor instructions as follows:



Object files are intermediate files that represent an incomplete copy of the program: each source file only expresses a piece of the program, so when it is compiled into an object file, the object file has some markers indicating which missing pieces it depends on.

The linker takes those object files and the compiled libraries of predefined code that they rely on, fills in all the gaps, and spits out the final program, which can then be run by the operating system (OS). The compiler and linker are just regular programs.

The step in the compilation process in which the compiler reads the file is called parsing. In C++, all these steps are performed ahead of time, before you start running a program. In some languages, they are done during the execution process, which takes time.

This is one of the reasons C++ code runs far faster than code in many more recent languages. C++ actually adds an extra step to the compilation process: the code is run through a preprocessor, which applies some modifications to the source code, before being fed to the compiler.

C++ Program Structure

Let us look at a simple code that would print the words Hello World.

Sample program

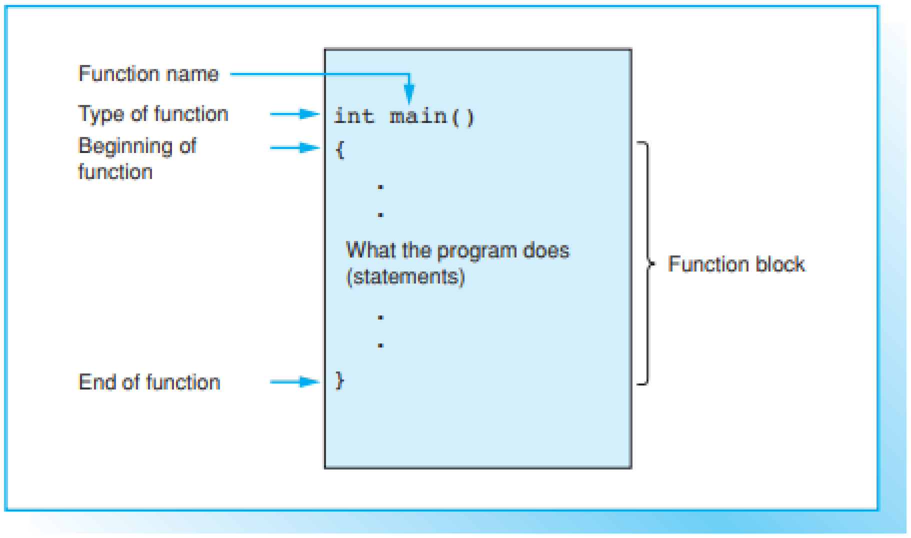
```
#include <iostream>
using namespace std;

int main()
{
   cout <<  "Enjoy yourself with C++!"  << endl;
   return 0;
}
```

Screen output

```
Enjoy yourself with C++!
```

Structure of function `main()`



Now let us look at the various parts of the above program:

1- Standard libraries section : **#include <iostream>**
   - **#include** is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
   - **The file <iostream>** which is a standard file that should come with the C++ compiler is short for input-output streams. This command contains code for displaying and getting an
     input from the user.
     Note: The Standard libraries section could have more than one standard file.

2- The line **using namespace std;** tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.

3- The next line '// **main()** is where program execution begins.' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.

4- The line **int main()** is the main function where program execution begins.

5- The next line **cout << "This is my first C++ program.";** causes the message "This is my first C++ program" to be displayed on the screen.

6- The next line **return 0;** terminates main() function and causes it to return the value 0 to the calling process.