Limitations of C Programming

1. C is said to be process oriented, structured programming language.
2. When program becomes complex, understating and maintaining such programs is very difficult.
3. Language don't provide security for data.
4. Using functions, we can achieve code reusability, but reusability is limited. The programs are not extendible.

Let's understand the differences between C and C++.

No.	C	C++
1)	C follows the **procedural style programming.**	C++ is multi-paradigm. It supports both **procedural and object oriented.**
2)	Data is less secured in C.	In C++, you can use modifiers for class members to make it inaccessible for outside users.
3)	C follows the **top-down approach.**	C++ follows the **bottom-up approach.**
4)	C does not support function overloading.	C++ supports function overloading.
5)	In C, you can't use functions in structure.	In C++, you can use functions in structure.
6)	C does not support reference variables.	C++ supports reference variables.
7)	In C, **scanf() and printf()** are mainly used for input/output.	C++ mainly uses stream **cin and cout** to perform input and output operations.
8)	Operator overloading is not possible in C.	Operator overloading is possible in C++.

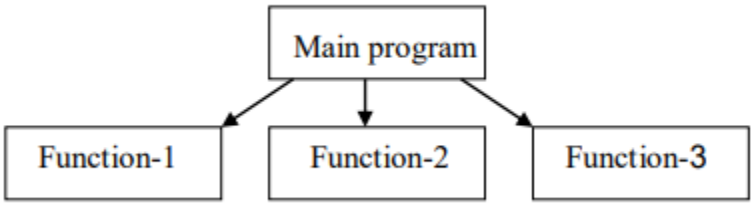
9)	C programs are divided into **procedures and modules**	C++ programs are divided into **functions and classes.**
10)	C does not provide the feature of namespace.	C++ supports the feature of namespace.
11)	Exception handling is not easy in C. It has to perform using other functions.	C++ provides exception handling using Try and Catch block.
12)	C does not support the inheritance.	C++ supports inheritance.

Classification of high-level Languages

1. Procedure Oriented Programming language (POP)
2. Object oriented programming languages (OOP)

Procedure Oriented Programming language (POP)

In the procedure oriented approach, the problem is viewed as sequence of things to be done such as reading , calculation and printing. Procedure oriented programming basically consist of writing a list of instruction or actions for the computer to follow and organizing these instruction into groups known as functions.



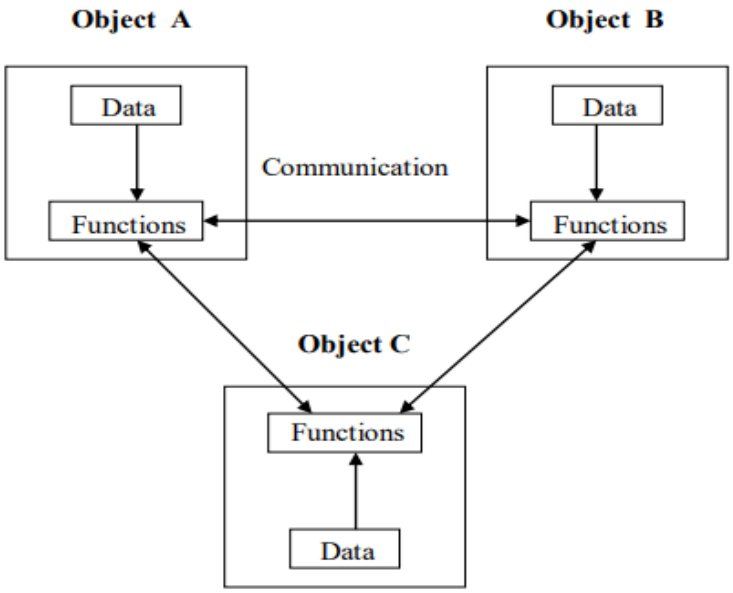
The disadvantage of the procedure-oriented programming languages is:

1. Global data access
2. It does not model real word problem very well

3. No data hiding

Object Oriented Programing

"Object oriented programming as an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand".



Features of the Object-Oriented programming

1. Emphasis is on doing rather than procedure
2. programs are divided into what are known as objects.
3. Data structures are designed such that they characterize the objects.
4. Functions that operate on the data of an object are tied together in the data structure.
5. Data is hidden and can't be accessed by external functions.
6. Objects may communicate with each other through functions.
7. New data and functions can be easily added.
8. Follows bottom-up approach in program design

History of C++

1. Inventor of C++ is Bjarne Stroustrup.
2. C++ is derived from C and simula
3. Its initial name was "C With Classes".

4. At is developed in "AT&T Bell Lab" in 1979.
5. It is developed on Unix Operating System.
6. In 1983 ANSI renamed "C With Classes" to C++.
7. C++ is objet oriented programming language

Why Use a Language Like C++?

Conciseness: programming languages allow us to express common sequences of commands more concisely. C++ provides some especially powerful shorthand's.

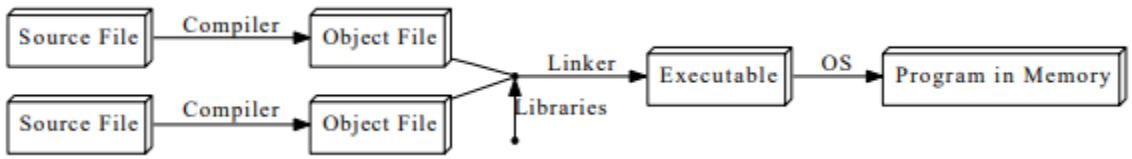
Maintainability: modifying code is easier when it entails just a few text edits, instead of rearranging hundreds of processor instructions. C++ is object oriented (more on that in Lectures 7-8), which further improves maintainability.

Portability: different processors make different instructions available. Programs written as text can be translated into instructions for many different processors; one of C++'s strengths is that it can be used to write programs for nearly any processor.

C++ is a high-level language: when you write a program in it, the shorthand's are sufficiently expressive that you don't need to worry about the details of processor instructions. C++ does give access to some lower-level functionality than other languages (e.g. memory addresses).

The Compilation Process

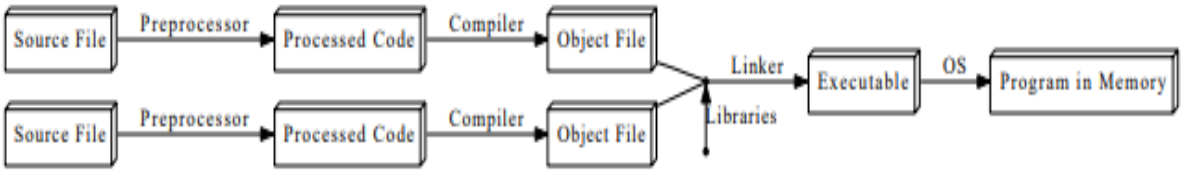
A program goes from text files (or source files) to processor instructions as follows:



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. Thus, the modified diagram is:



Hello World – First C++ Program

```
#include<iostream>
//Single line comment
using namespace std;
//This is where the execution of program begins
int main()
{
   // displays Hello World! on screen
   cout<<"Hello World!";

   return 0;
}
```

#include<iostream> – This statements tells the compiler to include iostream file. This file contains pre defined input/output functions that we can use in our program.

using namespace std; – A namespace is like a region, where we have functions, variables etc and their scope is limited to that particular region. Here std is a namespace name, this tells the compiler to look into that particular region for all the variables, functions, etc. I will not discuss this in detail here as it may confuse you. I have covered this topic in a separate tutorial with examples. Just follow the tutorial in the given sequence and you would be fine.

int main() – As the name suggests this is the main function of our program and the execution of program begins with this function, the int here is the return type

which indicates to the compiler that this function will return a integer value. That is the main reason we have a return 0 statement at the end of main function.

cout << "Hello World!"; – The cout object belongs to the iostream file and the purpose of this object is to display the content between double quotes as it is on the screen. This object can also display the value of variables on screen (don't worry, we will see that in the coming tutorials).

return 0; – This statement returns value 0 from the main() function which indicates that the execution of main function is successful. The value 1 represents failed execution.

C++ Basic Input/Output

C++ I/O operation is using the stream concept. Stream is the sequence of bytes or flow of data. It makes the performance fast.

If bytes flow from main memory to device like printer, display screen, or a network connection, etc, this is called as **output operation.**

If bytes flow from device like printer, display screen, or a network connection, etc to main memory, this is called as **input operation.**

I/O Library Header Files

Let us see the common header files used in C++ programming are:

Header File	Function and Description
<iostream>	It is used to define the **cout, cin and cerr** objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.
<iomanip>	It is used to declare services useful for performing formatted I/O, such as **setprecision and setw.**
<fstream>	It is used to declare services for user-controlled file processing.

Standard output stream (cout)

The **cout** is a predefined object of **ostream** class. It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream insertion operator (<<) to display the output on a console

Let's see the simple example of standard output stream (cout):

```
#include <iostream>
using namespace std;
int main( ) {
   char ary[] = "Welcome to C++  ";
   cout << "Value of ary is: " << ary << endl;
}
```

Standard input stream (cin)

The **cin** is a predefined object of **istream** class. It is connected with the standard input device, which is usually a keyboard. The cin is used in conjunction with stream extraction operator (>>) to read the input from a console.

Let's see the simple example of standard input stream (cin):

```
#include <iostream>
using namespace std;
int main( ) {
  int age;
   cout << "Enter your age: ";
   cin >> age;
   cout << "Your age is: " << age << endl;
}
```