Introduction to C++

# WHAT IS C++?

C++ evolved from its predecessor the C programming language and was developed by *Bjrane Stroustrup* in 1979 at *Bell Laboratory***.** Originally the language was called “*C with Classes*”. Since the language was created it has undergone three major revisions and the final draft was passed out on November 14, 1997 and an ANSI/ISO standard for C++ was passed in 1998.

As an object-oriented language, C++ is very powerful and has the extensibility to write large-scale programs. C++ is one of the most popular programming languages taught from school to the college level. Most of the programs you use on your PC every day are written in C++.

C++ has been certified as a 99.9% pure standard. This makes it a portable language. There is a C++ compiler for every major operating system, and they all support the same C++ language. (Some operating systems support extensions to the basic language, but all support the C++ core.)

# WHAT IS THE CONCEPT OF OBJECT ORIENTED PROGRAMMING?

When we say C++ is an object oriented programming paradigm we imply that in C++ we can make programs that can relate to the real world. Anything you think of – say a car is an object. We can say that the *object car* belongs to a *class of automobiles* and it would perform various *functions.* So if I had to define this class of automobiles (And it is the way I want to implement it in my program), it would look something like this:

Class automobile

{

*Type* engine;

*Type* make;

*Type* colour;

……..

……..

……..

Public:

*Type* function1();

*Type* function2();

……………………

……………………

……………………

}

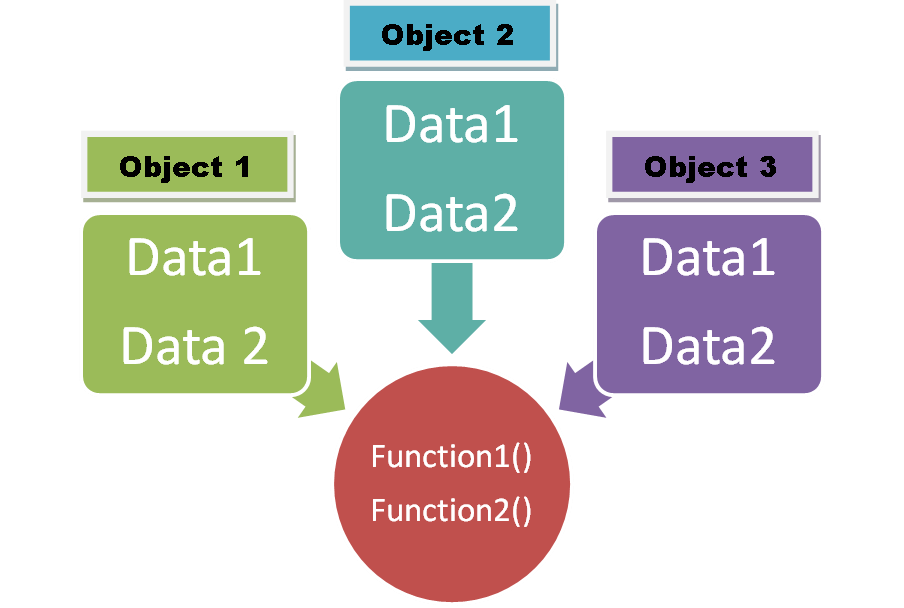
automobile car1,car2,car3;

Above we see a class automobile with various types of attributes such as engine, make, colour and various functions function1, function2 etc. Don’t worry about the TYPE in italics in the above code. We will come to that in the upcoming chapters.

For the above mentioned class we have created three objects car1, car2 and car3. So the objects car1, car2 and car3 will be *instances* of the class automobile. They will have their respective engine, make, colour attributes and each of the objects

will perform their tasks using the functions defined in the class *function1, function2* and so on using their respective variables (i.e. engine, make, colour etc.).

You can visualize the above explanation as follows:



Here object1, object 2 and object3 are instances of a class and function1(), function2() are the functions defined for the class.

# WHY OF OBJECT ORIENTED PROGRAMMING?

Object oriented programming helps implement the concept of:

* **ENCAPSULATION**

Encapsulation is wrapping up of all data and function into a single unit. It binds the code and the data that it manipulates and keeps it safe from outside interface and misuse. In OOP an object is the device that supports encapsulation.

We use data abstraction to implement data hiding. Abstraction is basically an act of representing essential features without including the background details. We implement abstraction in C++ using access specifiers such as public, private and protected. We will discuss these topics in the later section.

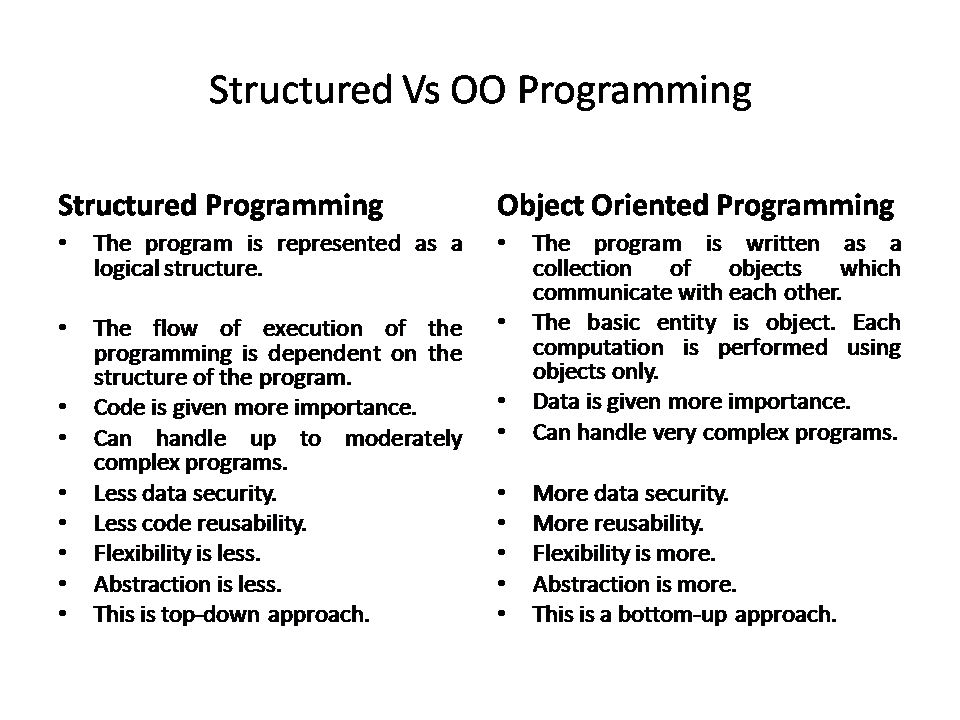
* **POLYMORPHISM**

Polymorphism simply can be put forth as : “one interface, multiple methods”. You can define multiple functions in C++ with the same name but make them behave separately depending upon the situation. All this might seem a bit confusing but we will see about this in detail in the sections that we cover later. For now just remember – Polymorphism enables programmers to deal in generalities and let the execution-time environment handle the specifics.

* **INHERITANCE**

Inheritance is by far the most interesting and important aspect of Object Oriented Programming. Inheritance helps us define real world relations between different classes and reflect them in the respective objects. For e.g. : We say that a child inherits some of their properties from their parents and has some properties on their own. Similarly in OOP when we say that a class inherits some other class we mean that the parent(base) class (the class being inherited) is sharing some of its properties with the child(derived) class(the class that is inheriting the base class). Again we will see this later in detail.

# ADVANTAGE OF OOP OVER STRUCTURED PROGRAMMING:



# SOME DEFINATIONS:

Function : A group of statements that are used to perform a set of specific tasks.

Class: Classes define types of data structures and the functions that operate on those data structures. They are just like blueprints to define an object.

Objects: An object is an instantiation of a class. It helps implement the blueprint provided by the class and build over it.