1.1 EVOLUTION OF C++

C++ is an object-oriented programming language and is considered to be an extension of C. Bjarne Stroustrup at AT&T Bell Laboratories, Murray Hill, New Jersey (USA) developed it in the early eighties of twentieth century. Stroustrup, a master of Simula67 and C, wanted to combine the features of both the languages into a more powerful language that could support object-oriented programming with features of C. The outcome was C++ as per Figure 1.1. Various ideas were derived from ***SIMULA67*** and ***ALGOL68.*** Stroustrup called the new language ***'C with classes'.*** However, in 1983, the name was changed to C++. The thought of C++ came from the C increment operator ++. Rick Mascitti coined the term C++ in 1983. Therefore, C++ is an extended version of C. C++ is a superset of C. All the concepts of C are applicable to C++ also.

For developing complicated applications object-oriented languages are most convenient and easy. Hence, a programmer must be aware of the new concepts of object-oriented techniques.

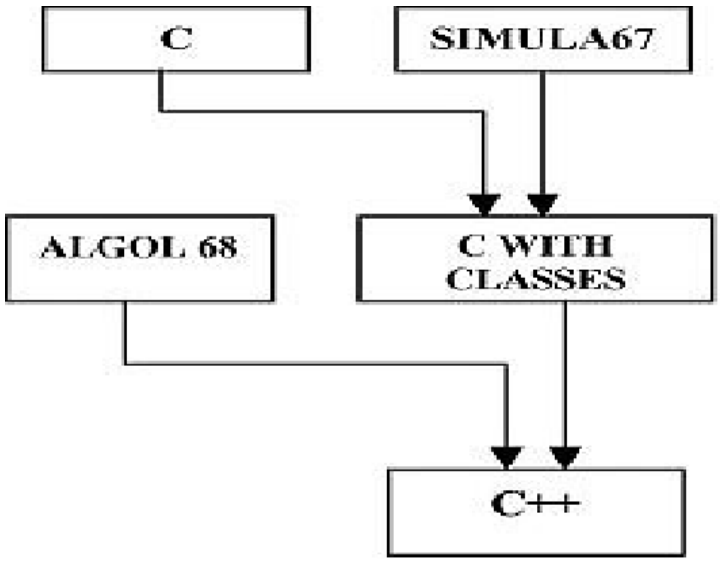


Fig.1

1.2 ANSI STANDARD

ANSI stands for American National Standards Institute. This Institute was founded in ***1918.*** The main goal for establishing this Institute was to suggest, reform, recommend, and publish standards for data processing in the USA. This committee sets up standards for the computer industry.

The recognized council working under the procedure of the American National Standard Institute (ANSI) has made an international standard for C++. The C++ standard is also referred to as ISO (International Standard Organization) standard. The process of standardization is gradual and the first draft of the planned ANSI standard was made on ***25 January 1994.*** This book will continue to refer to ANSI standard code, which is more commonly used. The ANSI standard is an attempt to ensure that C++ is portable.

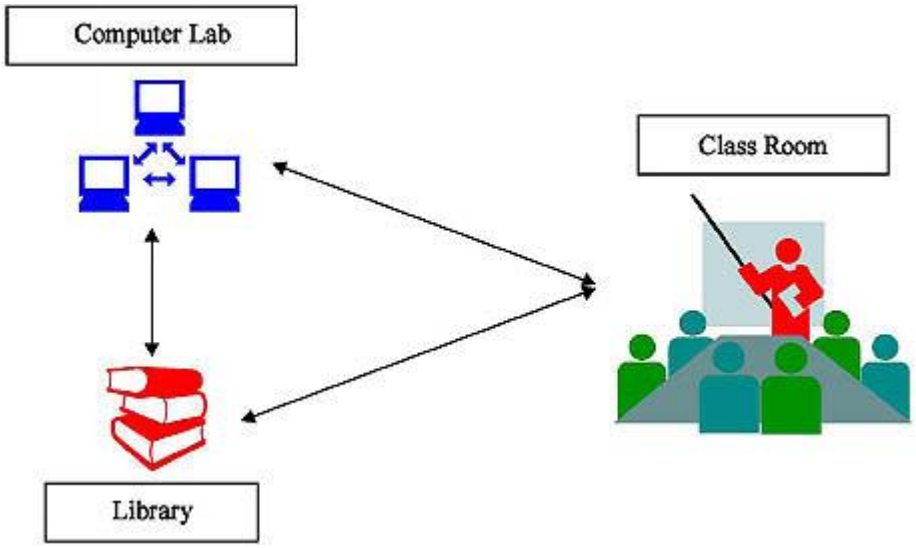
1.3 OBJECT-ORIENTED TECHNOLOGY

Nature is composed of various objects. Living beings can be categorized with different objects.

Let us consider an example of a teaching institute having two different working sections. They are teaching and non-teaching. Further sub-grouping of teaching and non-teaching can be made for the ease of management. The various departments of any organization can be thought as objects working with certain goals and objectives.

Usually, an institute has faculties for different departments. Also, laboratory staff for the assistance in conducting of practicals and site development section for beautification of the campus are essential. The Principal is a must for the overall management of the Institute. The accounts department is also required for handling monetary transactions, and salaries of the employees. The sports section is entrusted the responsibility of sports activities, the Registrar for administration and his staff for dealing with administrative-matters of the Institute and so on are required. Each department has an In-Charge who has clear-cut responsibilities. Every department has its own work as stated above. When the work is distributed to different departments as shown in Figure 1.2, it becomes easy to accomplish goals and objectives. The activities are carried out smoothly. The burden of one particular department has to be shared among personnel. The staff in the department are controlled properly and act according to the instructions laid down by the management. The faculty performs activities related to teaching. If higher authority needs to know the details regarding the theory, practical, seminar and project workload of individuals of the department then some responsible person from the department furnishes the same. This way some responsible person from the department accesses the data and provides the higher authority with requisite information. Only authorized persons have access to data base.

As shown in Figure 1.2 an institute is divided into different departments such as library, classroom, computer laboratory, etc. Each department performs its own activities in association with other departments. This theory of objects can be extended to every walk of life and can be implemented with the help of software. In general, objects are in terms of entities.



1.4 DISADVANTAGE OF CONVENTIONAL PROGRAMMING

Traditional programming languages such as COBOL, FORTRAN etc. are commonly known as procedure oriented languages. The program written in these languages consists of a sequence of instructions that tells the compiler or interpreter to perform a given task. When program code is large then it becomes inconvenient to manage it. To overcome this problem, procedures or subroutines were adopted to make a program more understandable to the programmers. A program is divided into many functions. Each function can call another function as shown in Figure1.3. Each function has its own task. If the program is too large the function also creates problems. In many programs, important data variables are declared as global. In case of programs containing several functions, every function can access the global data as simulated in Figure 1.4. Generally in a program of large size, it becomes difficult to understand the various data used in different functions. As a result, a program may have several logical errors.

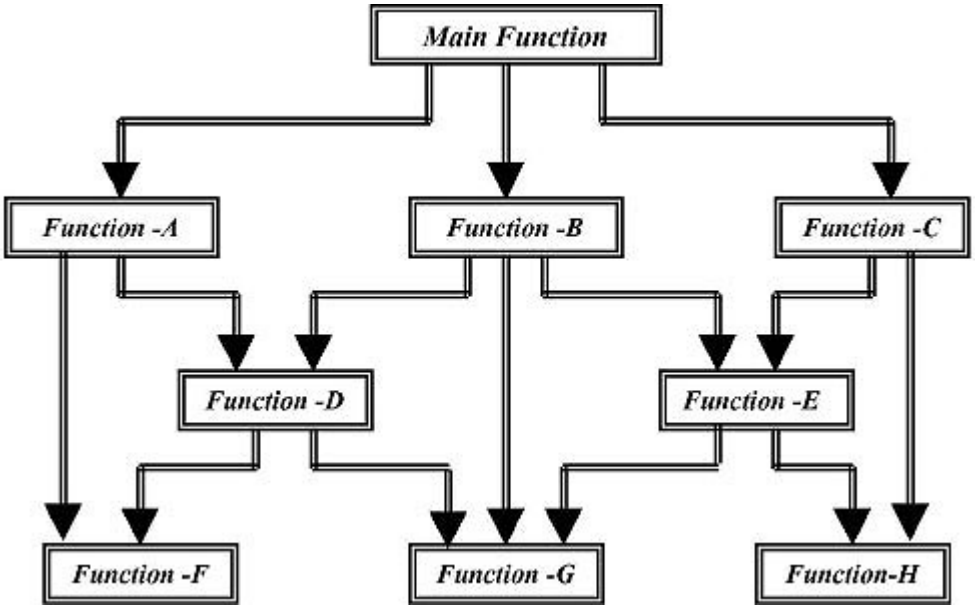
Following are the drawbacks observed in monolithic, procedural, and structured programming languages:

(1) Large size programs are divided into smaller programs known as functions. These functions can call one another. Hence, security is not provided.

(2) Importance is not given to security of data but on doing things.

(3) Data passes globally from one function to another.

(4) Most functions have access to global data.



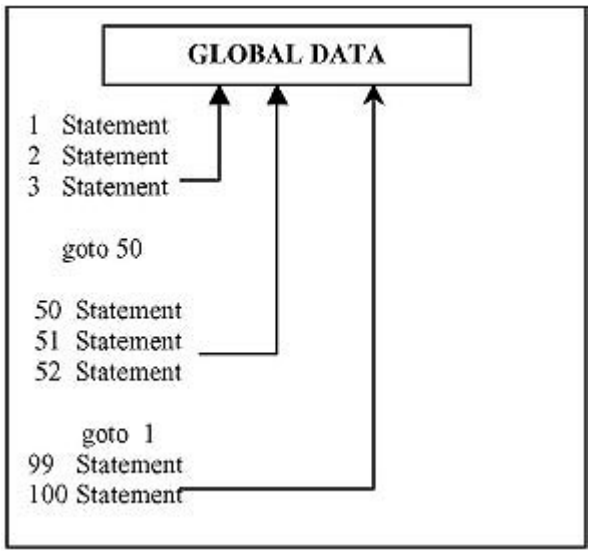
1.5 PROGRAMMING PARADIGMS

(1) Monolithic Programming

(A) In monolithic programming languages such as BASIC and ASSEMBLY, the data variables declared are global and the statements are written in sequence.

(B) The program contains jump statements such as goto, that transfer control to any statement as specified in it. Figure 1.5 shows a program of monolithic type. The global data can be accessed from any portion of the program. Due to this reason the data is not fully protected.

(C) The concept of subprograms does not exist and hence useful for smaller programs.



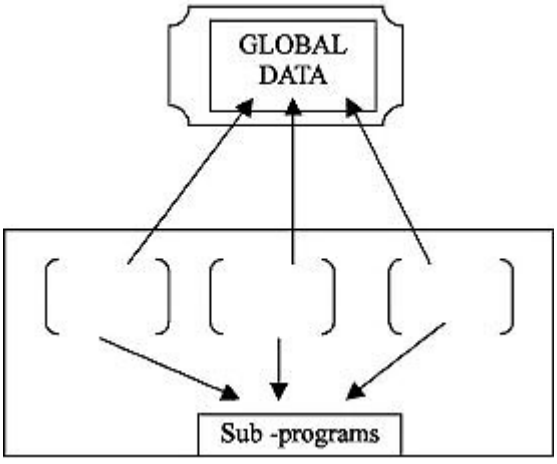
(2) Procedural Programming

(A) In procedural programming languages such as FORTRAN and COBOL, programs are divided into a number of segments known as subprograms. Thus it focuses on functions apart from data. Figure 1.6 describes a program of procedural type. It shows different subprograms having access to the same global data. Here also, the data is not fully protected.

(B) The control of program is transferred using unsafe goto statement.

(C) Data is global and all the subprograms share the same data.

(D) These languages are used for developing medium size applications.



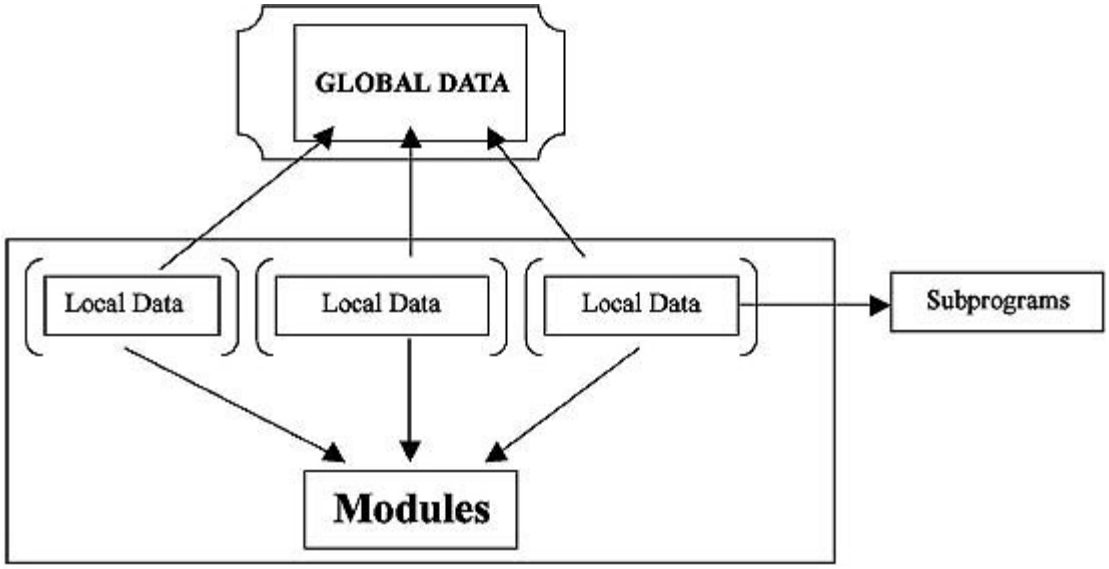
(3) Structured Programming

(A) In structured programming languages such as PASCAL and C larger programs are developed. These programs are divided into multiple submodules and procedures.

(B) Each procedure has to perform different tasks.

(C) Each module has its own set of local variables and program code as shown in Figure 1.7. Here, the different modules are shown accessing the global data.

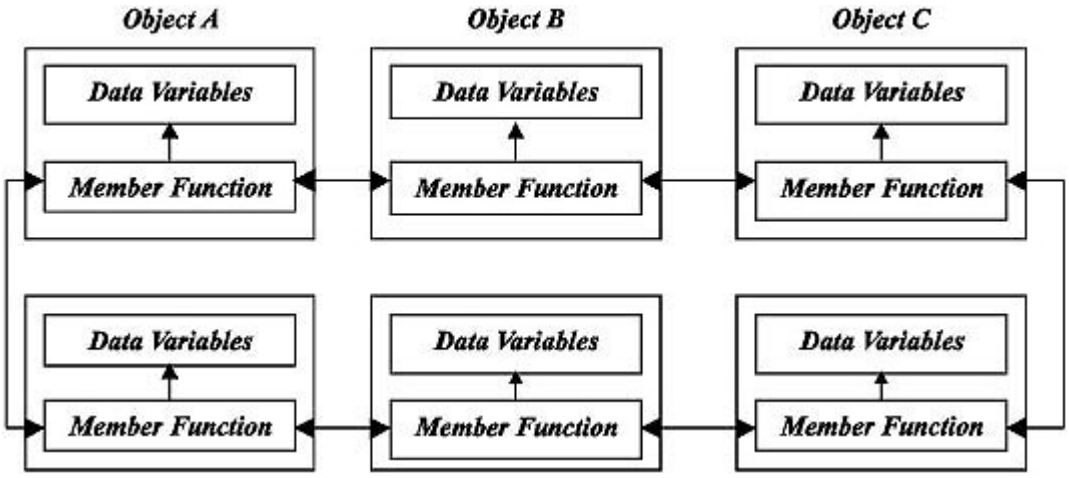
(D) User-defined data types are introduced.



1.6 PREFACE TO OBJECT-ORIENTED PROGRAMMING

The prime factor in the development of object-oriented programming approach is to remove some of the shortcomings associated with the procedure oriented programming. OOP has data as a critical component in the program development. It does not let the data flow freely around the systems. It ties data more firmly to the functions that operate on it and prevents it from accidental change due to external functions. OOP permits us to analyze a problem into a number of items known as objects and then assembles data and functions around these items as shown in Figure 1.8. The basic objective of OOP is to treat data and the program as

individual objects. Following are the important characteristics of object-oriented programming:



Highlights of OOP approach :

(1) OOP pays more importance to data than to function.

(2) Programs are divided into classes and their member functions.

(3) New data items and functions can be comfortably added whenever essential.

(4) Data is private and prevented from accessing external functions.

(5) Objects can communicate with each other through functions.

The OOP language is divided into four major concepts.

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

Encapsulation in C++

Encapsulation is one of the major cores of OOP. It is the mechanism that binds code and data together and keeps them safe and away from outside inference and misuse. In short, it isolates a data from outside world. Encapsulation can be called as the core concept of OOP because it is encapsulation that reduces the maintenance burden, and limiting your exposure to vulnerabilities. The

programmer has to design a well-defined interface to control the access of a particular code and data. A class defines the data and code which is shared by a set of objects. A class is a logical construct and an object is a physical construct which exists in reality that is like a chair or a table. A class comprises of code and data which are collectively called members of the class.

Inheritance in C++

It is the technique used in OOP that one object acquires the properties of another object without redefining in order to create well defined classes. This technique also supports the hierarchical classification in which each object would not need to be redefined by all its characteristics explicitly. It prevents the programmer from unnecessary work and it is the inheritance concept that does this job. Inheritance gives us the facility to an object to inherit only those qualities that make it unique within its class. In general, it can inherit any attributes from its parent. Again to be well noted that it is the inheritance that get both data and functionality. We can later add any new data and methods needed. Different names have been used in different times for a parent class and a child class. The concept of parent and child class was developed to manage generalization and specialization in OOP and it is represented by a is-a relationship. But now-a-days, the following terms in object oriented programming is commonly used names given to parent and child class.

- Super Class: Parent Class
- Sub Class: Child Class
- Base Class: Parent Class
- Derived Class: Child Class

The generalization means that an object encapsulates common state behavior for a category of objects. The concept of inheritance makes code readable and avoids repetition. Suppose we have different shapes and they have same or different colors. With the concept of inheritance included later on to design a shape class that has a color attribute and to inherit the color attributes.

C++ also supports multiple inheritances. It is the process that a child class can have multiple parent classes. In other words, it occurs when a class is derived from more than one classes. A good example for a better understanding in a real life would be a child with characteristics of his parents. For a novice programmer it is not only difficult to handle multiple inheritances but also allows ambiguity for the compiler. There are programming languages such as Java and the .NET that does not support

multiple inheritances.

Polymorphism in C++

Photo Credit: j2sejava.wordpress.com

This concept was first identified by Christopher Strachey in 1967. Later, developed by Hindley and Milner. It is not important to understand how polymorphism is implemented but how it works. C++ offers two ways to retrieve the type of objects and expressions while the program is running, at compile time and run time. The basic idea behind polymorphism is that the compiler does not know which function to call at compile-time. This is done at run time. The term polymorphism refers to the fact that it is used in object oriented programming that the decision to be used will not be known at compiler time. The decision to be postponed until the program is actually executed which respond at run time. The run time is referred to late binding. Polymorphism is realized by this feature.

Class :-

A class is basically a collection of related data items and the function operating on them referenced under one name.

Syntax :

```
class class-name {
        data-item1;
        data-item2;
        data-item3;
public :
    void fucntion1();
    void function2();
}object1,object2;
```

Example :

```
class account {
        int acno;
        char type;
        float balance;
        float deposit(float amount)
        {
            balance=balance+amount;
        }
```

```
float withdraw(float amount)
{
        balance=balance-amount;
        return balance;
}
```

Practice Problems :

1. Write a program using a class to store price list of 5 items and to print the largest price as well as the sum of all prices.
2. Define a class named Admission with ad_no, name, class and fees as the data items. This should include function read_data(), print_data() and draw_nos() to choose 2 students randomly and print their details. Hint : Use the randomize() and random functions to generate a random number.