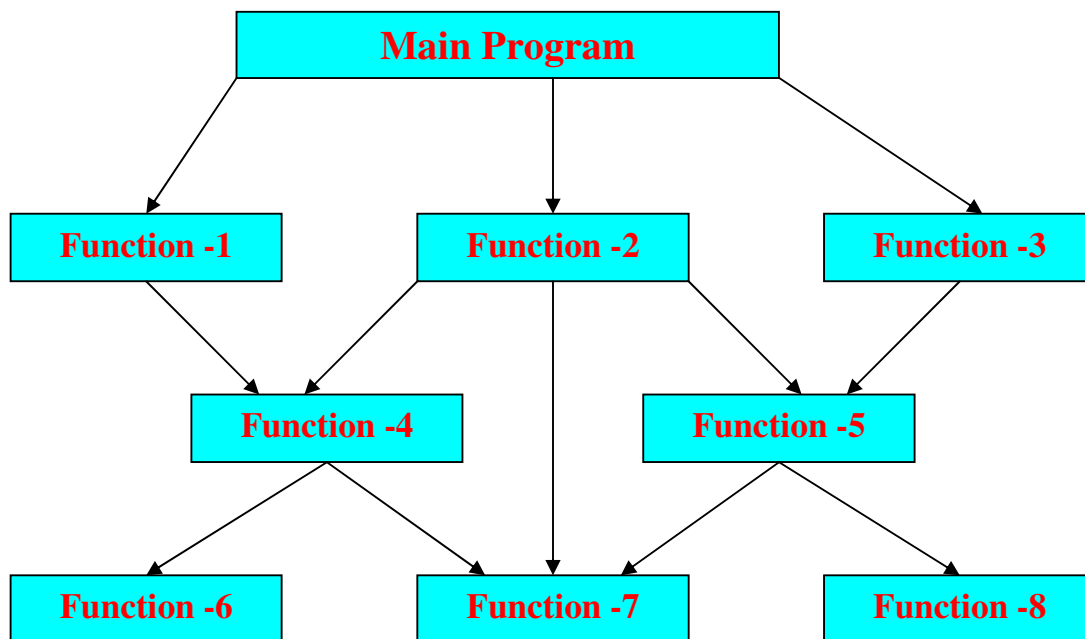


# 1. Principles of Object – Oriented Programming

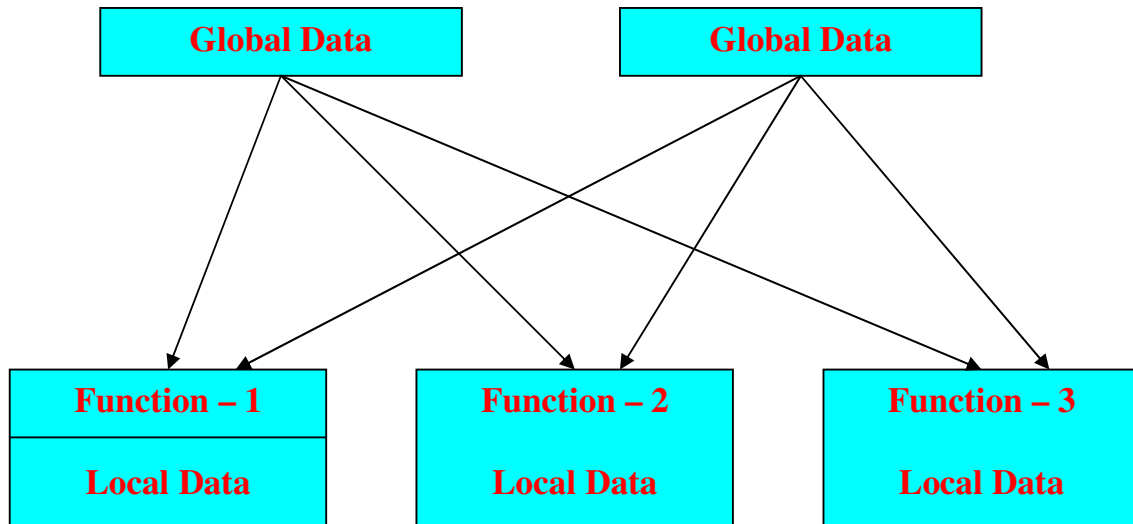
## ❖ Procedure Oriented Programming

- Conventional programming, using high level languages such as COBOL, FORTRAN and C is commonly known as procedure – Oriented programming - (P O P). In the procedure – oriented approach, the problem is viewed as a sequence of things to be done such as reading, calculating and printing. A number of functions are written to accomplish these tasks. The primary focus is on functions. A typical program structure for procedure programming is shown in following figure. The technique of hierarchical decomposition has been used to specify the tasks to be completed for solving a problem.



### Typical structure of Procedure - Oriented Program

- Procedure – Oriented programming basically consists of writing a list of instructions or actions to be performed for the computer to follow and organizing these instructions into groups known as functions. We normally use a flowchart to organize these actions and represent the flow of control from one action to another. While we concentrate on the development of functions, very little attention is given to the data that are being used by various functions. What happens to the data? How are they affected by the functions that work on them?
- In a multi-function program, many important data items are placed global so that they may be accessed by all the functions. Each function may have its own local data. Following figure shows the relationship of data and functions in a procedure-oriented program.



### Relationship of data and functions in procedural programming

- Global data are more vulnerable to an inadvertent change by a function. In a large program it is very difficult to identify what data is used by which function. In case we need to revise an external data structure, we also need to revise all functions that access the data. This provides an opportunity for bugs to creep in.
- Another serious drawback with the procedure approach is that it does not model real world problems very well. This is because functions are action- oriented and do not really corresponding to the elements of the problem.



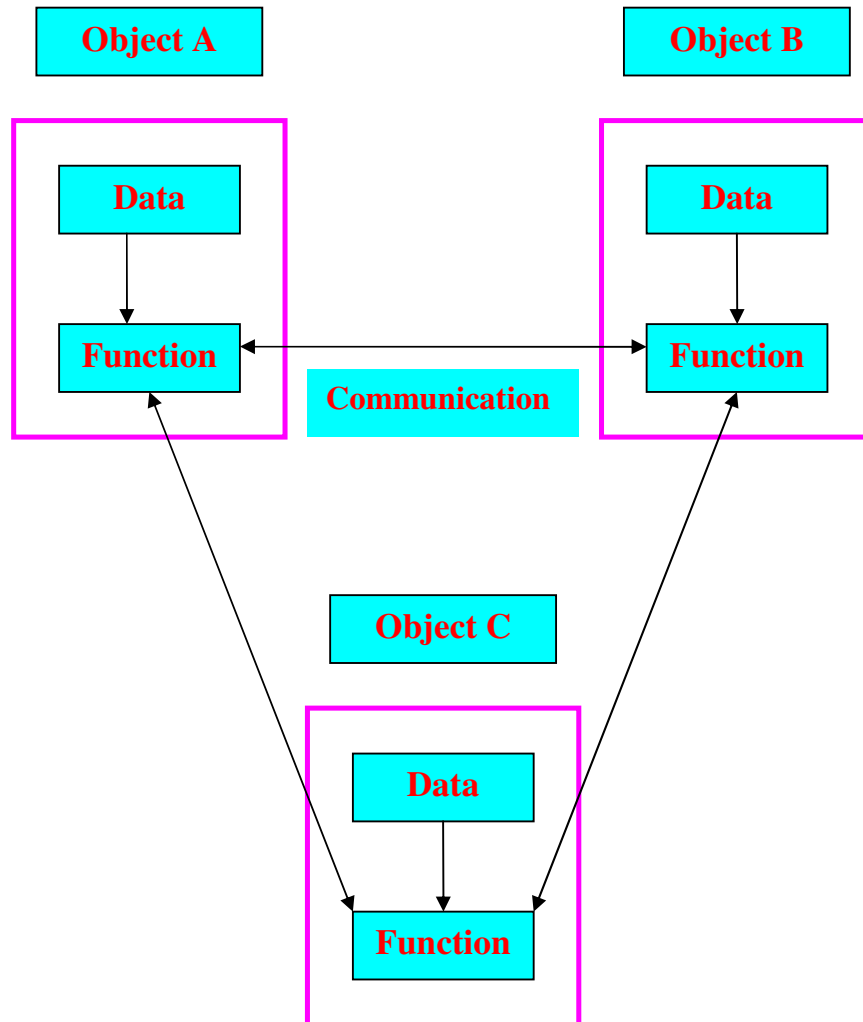
### Some characteristics exhibited by procedure-oriented programming are described as follows.

1. Emphasis is on doing things – algorithms.
2. Large programs are divided into smaller programs known as functions.
3. Most of the functions share global data.
4. Data move openly around the system from function to function.
5. Functions transform data from one form to another.
6. Employs top-down approach in program design.

### ❖ Object - Oriented Programming Paradigm

The major motivating factor in the invention of object – oriented approach is to remove some of the flaws encountered in the procedural approach. OOP treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions. OOP allows decomposition of a problem into a number of entities called objects

and then builds data and functions around these objects. The organization of data and functions in object-oriented programs is shown in following figure. The data of an object can be accessed only by the functions associated with that object; however, functions of one object can access the functions of other objects.



### Organization of Data and Functions in OOP



**Some of the striking features of object - oriented programming are described as follows.**

1. Emphasis is on data 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 cannot be accessed by external functions.

6. Objects may communicate with each other through functions.
7. New data and functions can be easily added whenever necessary.
8. Follows bottom-up approach in program design.

Object - Oriented programming is the most recent concept among programming paradigms and still means different things to different people. It is therefore important to have a working definition of object – oriented programming before we proceed further. We define

**“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.”**

Thus an object is considered to be a partitioned area of computer memory that stores data and set of operations that can access that data. Since the memory partitions are independent, the objects can be used in a variety of different programs without modifications.

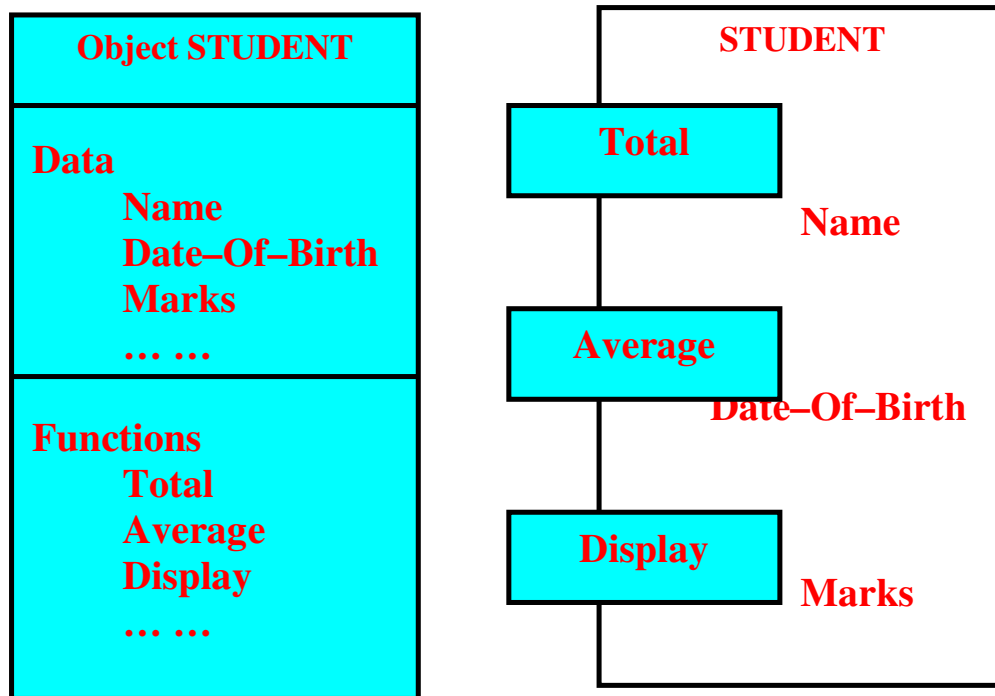
### ❖ **Basic concept of Object - Oriented Programming**

It is necessary to understand some of the concepts used extensively in object-oriented programming.

- 🌐 **Objects**
- 🌐 **Classes**
- 🌐 **Data abstraction and Encapsulation**
- 🌐 **Inheritance**
- 🌐 **Polymorphism**
- 🌐 **Dynamic binding**
- 🌐 **Message passing**

- 🌐 **Objects :** Objects are the basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent the user defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in memory and have an associated address like a record in Pascal or a structure in C.

When a program is executed, the objects interact by sending messages to one another. For example, if “customer” and “account” are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance. Each object contains data, and code to manipulate the data. Objects can interact without having to know details of each other’s data or code. It is sufficient to know the type of message accepted and the type of response returned by the objects. Although different authors represent them differently, following figure shows two notations that are popularly used in object – oriented analysis and design.



## Two ways of representing an object

- **Classes :** Objects contain data and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects of similar type. For example, mango, apple, and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language. The syntax used to create an object is no different than the syntax used to create an integer object in C. If fruit has been defined as a class, then the statement

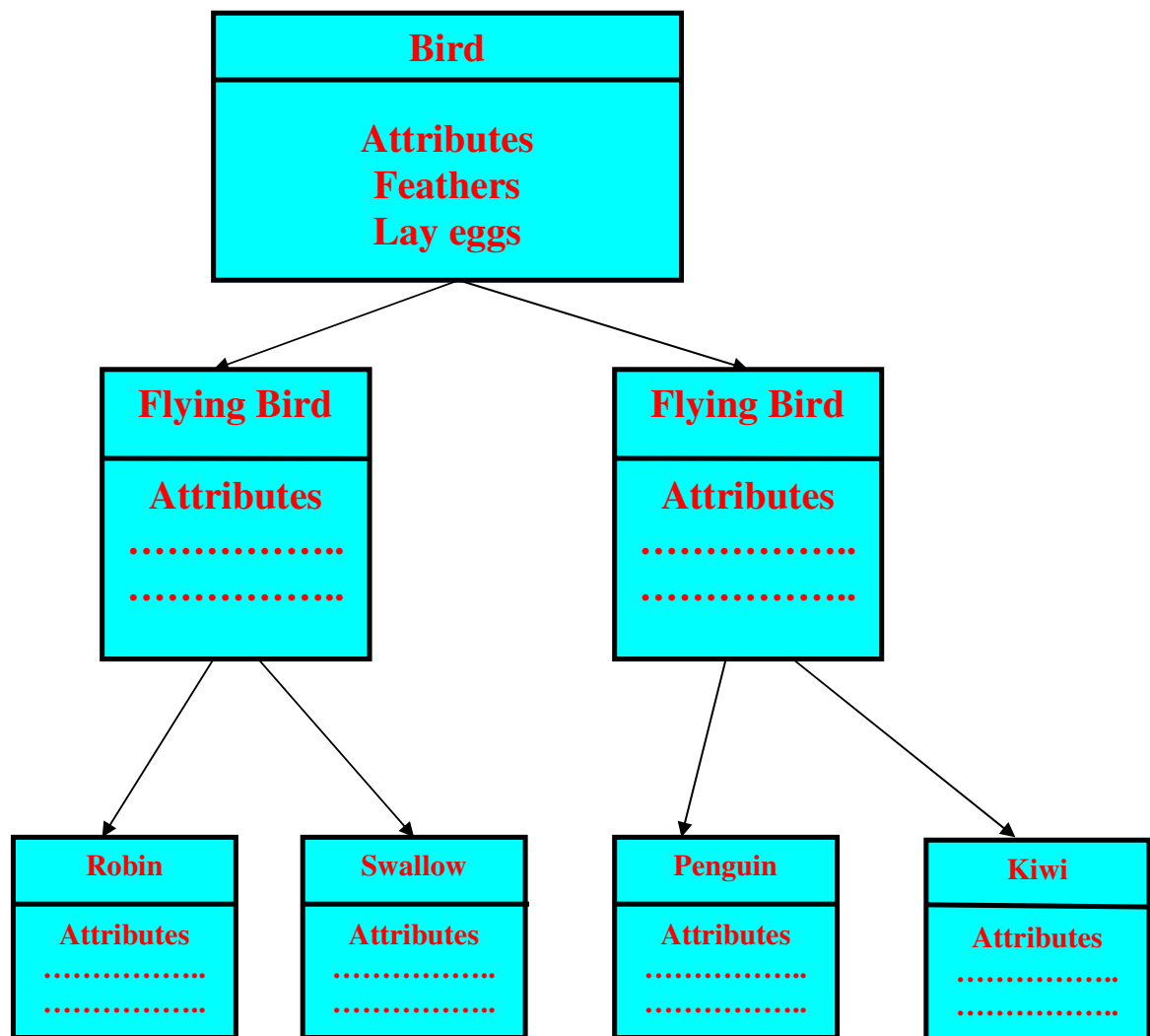
**fruit mango;**

will create an object mango belonging to the class fruit.

- **Data Abstraction and Encapsulation :** The wrapping up of data and functions into a single unit called class is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called data hiding or information hiding. Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes. They encapsulate all the essential properties of the objects that are to be created. The attributes are sometimes called data members because they hold information. The functions that operate on these data are sometimes called


methods or member functions. Since the classes use the concept of data abstraction, they are known as Abstract Data Types – ADT.

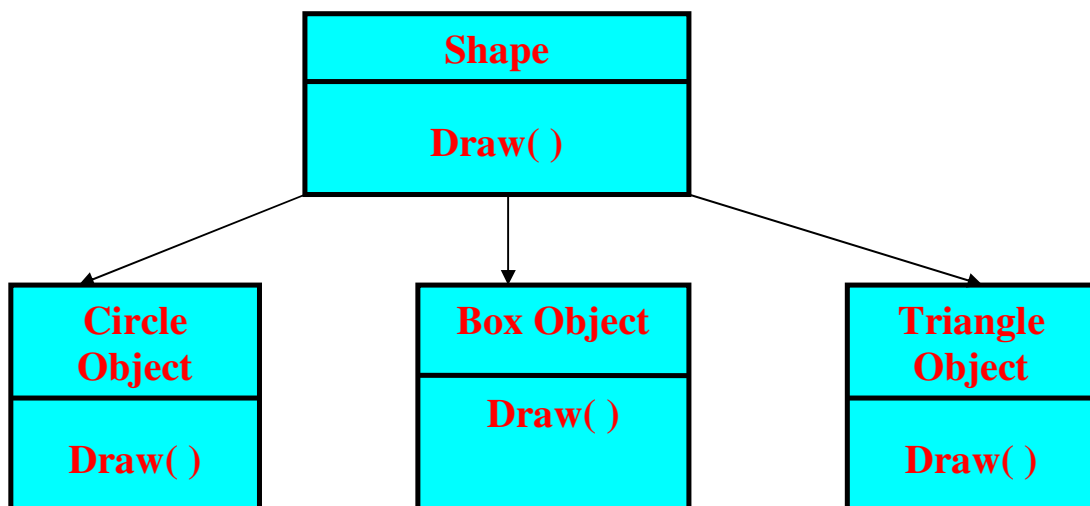
- 🌐 **Inheritance** : Inheritance is the process by which objects of one class acquire the properties of another class. It supports the concept of hierarchical classification. For example, the bird 'robin' is a part of the class 'flying bird' which is again a part of the class 'bird'. The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived as following figure.



### Property Inheritance

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both the classes. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost but not exactly what he or she wants and to tailor the class in such a way that it does not introduce any undesirable side effects into the rest of the classes. Note from the figure that each sub-class defines only those features that are unique to it. Without the use of classification each class would have to explicitly include all of its features.

 **Polymorphism :** Polymorphism is another OOP concept. Polymorphism a Greek term means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation. For example consider the operation of addition. For two numbers the operation will generate a sum. If the operands are string then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instances is known as operator overloading. Following figure illustrate that a single function name can be used to handle different number and different types of arguments. This is something similar to a particular word having several different meaning depending on the context. Using a single function name to perform different types of tasks is known as function overloading. Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ. Polymorphism is extensively used in implementing inheritance.



### Polymorphism

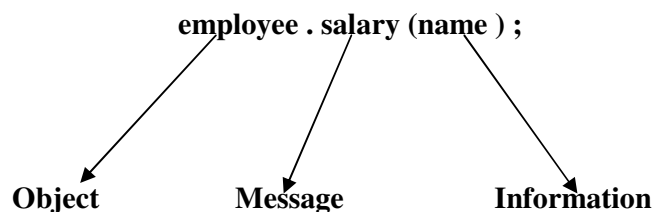
● **Dynamic Binding :** Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding also known as late binding means that the code associated with a give procedure call is not known until the time of the function call at run time. It is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depends on the dynamic type of that reference. Consider the procedure “draw” in above figure. By inheritance every object will have this procedure. Its algorithm is however unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time the code matching the object under current reference will be called.

● **Message Passing :** An object-oriented program consists of a set of objects that communicate with each other. The process of programming in object-oriented language involves the following basic steps :

1. Creating classes that define objects and their behavior.
2. Creating objects from class definitions and
3. Establishing communication among objects.

■ Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another. The concept of message passing makes it easier to talk about building systems that directly model or simulate their real-world counterparts.

■ A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired result. Message passing involves specifying the name of the object, the name of the function – message and the information to be sent. For example



■ Objects have a life cycle. They can be created and destroyed. Communication with an object is feasible as long as it is alive.

### ❖ Benefits of OOP

■ OOP offers several benefits to both the program designer and the user.



- Object-orientation contributes to the solution of many problems associated with the development and quality of software products.
- The new technology promises greater programmer productivity better quality of software and lesser maintenance cost.

**The principal advantages are as follows :**

- Through inheritance we can eliminate redundant code and extend the use of existing classes.
- We can build programs from the standard working modules that communicate with one another rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- It is possible to map objects in the problem domain to those in the program.
- It is to partition the work in a project based on objects.
- The data-centered design approach enables us to capture more details of a model in implementable form.
- Object-Oriented systems can be easily upgraded from small to large systems.
- Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

While it is possible to incorporate all these features in an object-oriented system, their importance depends on the type of the project and the preference of the programmer. There are a number of issues that need to be tackled to reap some of the benefits stated above. For instance, object libraries must be available for reuse. The technology is still developing and current products may be superseded quickly. Strict controls and protocols need to be developed if reuse is not to be compromised.

**Developing a software that is easy to use makes it hard to build. It is hoped that the object-oriented programming tools would help manage this problem.**

### **❖ Applications of Object – Oriented Programming**

OOP has become one of the programming buzzwords today. There appears to be a great deal of excitement and interest among software engineers in using OOP.

Applications of OOP are beginning to gain importance in many areas. The most popular application of object-oriented programming, up to now has been in the area of user interface design such as windows. Hundreds of windowing systems have been developed using the OOP techniques.

Real business systems are often much more complex and contain many more objects with complicated attributes and methods. OOP is useful in these types of applications because it can simplify a complex problem. The promising areas for application of OO include :

- 1. Real-time systems**
- 2. simulation and modeling**
- 3. Object-Oriented databases**
- 4. Hypertext, Hypermedia and experttext**
- 5. AI and Expert Systems**
- 6. Neural Networks and Parallel Programming**
- 7. Decision support and office automation system**
- 8. CIM / CAM / CAD system**

The richness of OOP environment has enabled the software industry to improve not only the quality of software systems but also its productivity. Object-Oriented technology is certainly changing the way the software engineers think, analyze, design and implement systems.

## ❖ What is C++ ?

C++ is an object – Oriented programming language. It was developed by Bjarne Stroustrup at AT & T Bell Laboratories in Murray Hill, New Jersey, USA, in the early of 1980s. Stroustrup, an admirer of simula67 and strong supporter of C, wanted to combine the best of both the languages and create a more powerful language that could support object-oriented programming features and still retain the power and elegance of C. The result was C++. Therefore, C++ is an extension of C with a major addition of the class construct feature of simula67. Since the class was a major addition to the original C language Stroustrup initially called the new language ‘C with classes’. However, later in 1983, the name was changed to C++. The idea of C++ comes from the C Increment operator ++, thereby suggesting that C++ is an augmented – incremented version of C. During the early 1990’s the language underwent a number of improvements and changes. In November 1997, the ANSI / ISO standard committee standardized these changes and added several new features to the language specifications.

C++ is a superset of C. Most of what we already know about C applies to C++ also. Therefore, almost all C programs are also C++ programs. However, there are a few minor differences that will prevent a C program to run under C++ compiler.

The most important facilities that C++ adds on to C are Classes, Inheritance, function overloading and operator overloading. These features enable creating of abstract data types inherit properties from existing data types and support polymorphism, therefore making C++ a truly object-oriented language.

The Object – Oriented features in C++ allow programmers to build large programs with clarity, extensibility and ease of maintenance, incorporating the spirit and efficiency of C. The addition of new features has transformed C from a language that currently facilitates top-down structured design to one that provide bottom – up Object – Oriented design.

## ❖ Applications of C++

C++ is a versatile language for handling very large programs. It is suitable for virtually any programming task including development of editors, compilers, databases, communication systems and any complex real-life application systems.

- 🌐 Since C++ allows us to create Hierarchy-related objects, we can build special object oriented libraries which can be used later by many programmers.
- 🌐 While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine-level details.
- 🌐 C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add the existing structure of an object.
- 🌐 It is expected that c++ will replace C as a general – purpose language in the near future.

## ❖ Input Output Operators

### 🌐 Input Operator

The statement

**cin >> number ;**

is an input statement and causes the program to wait for the use to type in a number. The number keyed in is place in the variable numebr1. The identifier **cin** – pronounced ‘C in’ – is a predefined object in C++ that corresponds to the standard input stream. Here, this stream represents the keyboard.

The operator >> is known as extraction or get from operator. It extracts or takes the value from the keyboard and assigns it to the variable on its right as following figure. This corresponds to the familiar scanf () function operation. Like <<, the operator >> can also be overloaded.

### 🌐 Output Operator

The statement

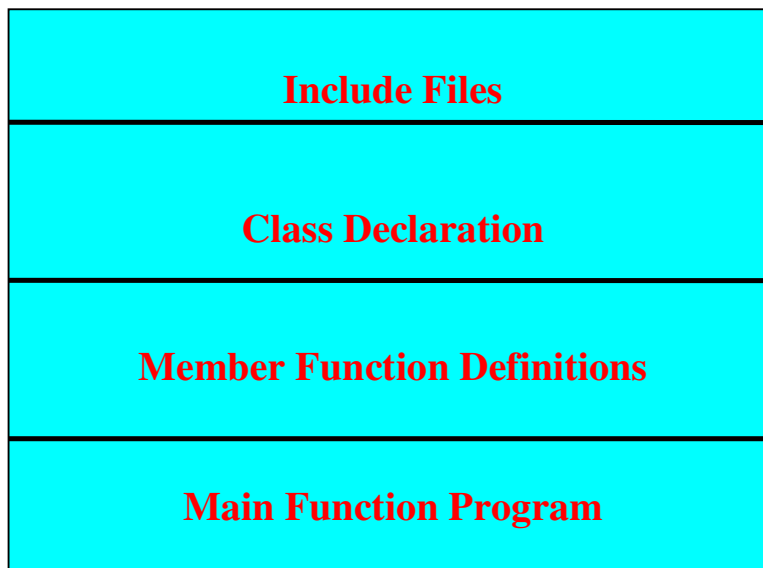
**cout << “C++ is better C.”;**

causes the string in quotation marks to be displayed on the screen. This statement introduces two new C++ features, cout and <<. The identifier cout – pronounced as ‘C Out’ – is a predefined object that represents the standard output stream in C++. Here the standard output stream represents the screen. It is also possible to redirect the output to other output devices.

The operator << is called the insertion or put to operator. It inserts or sends the contents of the variable on its right to the object on its left as following figure.

### ❖ Structure of C++ Program

A typical C++ Program would contain four sections as shown in following figure. These sections may be placed in separate code files and then compiled independently or jointly.



### Structure of a C++ program

It is a common practice to organize a program into three separate files. The class declarations are placed in a header file and the definitions of member functions go into another file. This approach enables the programmer to separate the abstract specification of the interface – class definition – from the implementation details – member functions definition. Finally, the main program that uses the class is placed in third file which “includes” the previous two files as well as any other files required.

This approach is based on the concept of client-server model shown as following figure. The class definition including the member functions constitute the server that provides services to the main program known as client. The client uses the server through the public interface of the class.