Unit 1

Introduction to Object Oriented Programming: Computer programming background- C++ overview. First C++ Program -Basic C++ syntax, Object Oriented Programming: What is an object, Classes, methods and messages, abstraction and encapsulation, inheritance, abstract classes, polymorphism?

Procedure Oriented Programming Language

- 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.**
- A typical structure for procedural programming is shown
- in fig. below

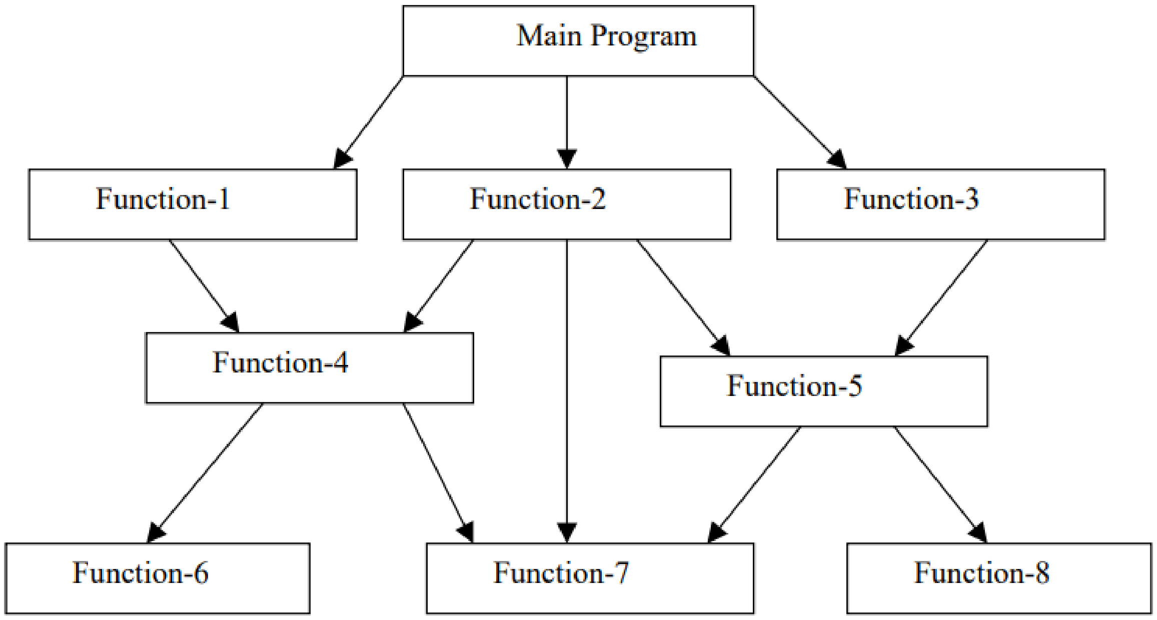
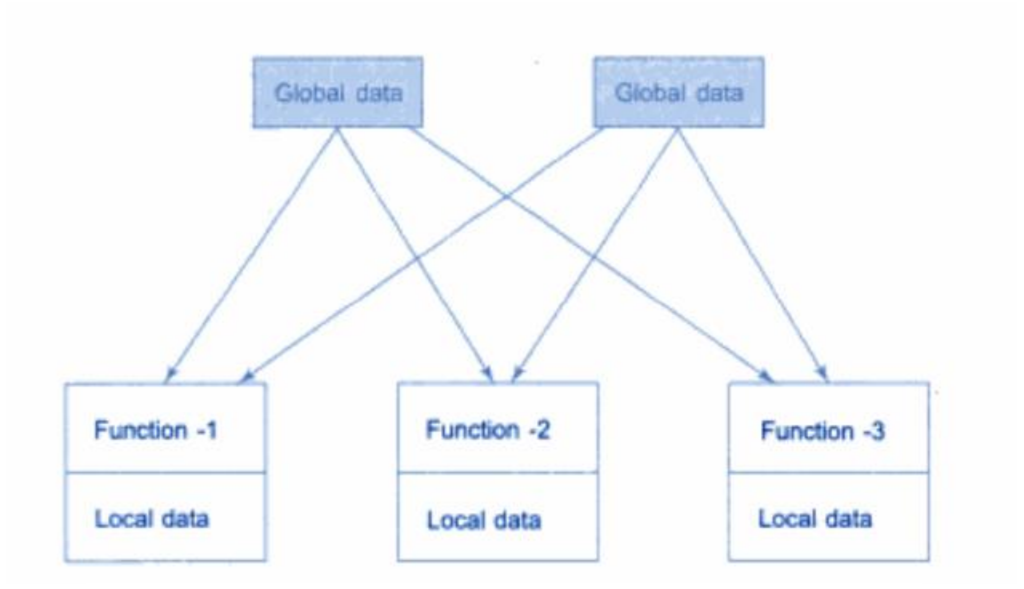


Fig. 1.2 Typical structure of procedural oriented programs

- The technique of hierarchical decomposition has been used to specify the tasks to be completed for solving a problem.
- In a multi-function program, many important data items are placed as global so that they may be accessed by all the functions. Each function may have its own local data.
- 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 procedural approach is that we do not model real world problems very well



Relationship of data and the function In Procedural oriented

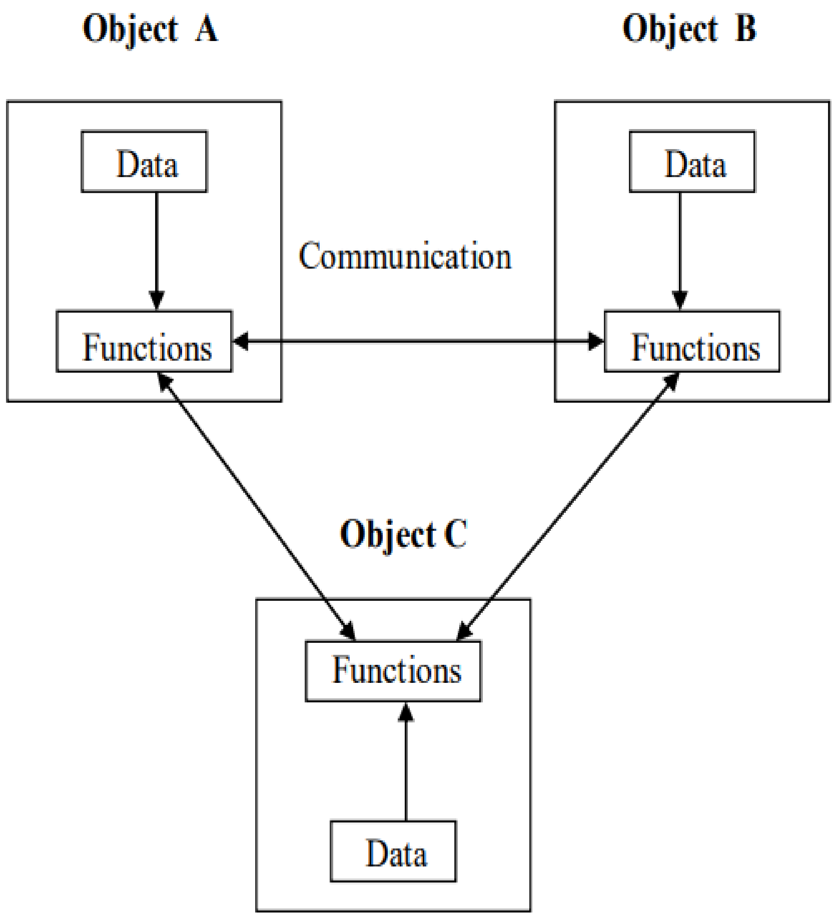
Some Characteristics exhibited by procedure-oriented programming are:

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

Object Oriented Paradigm

- **Object-oriented programming (OOP) is a programming paradigm that organizes code around objects, which are instances of classes that encapsulate data and behavior. It is a widely used programming approach that provides a way to structure and design software systems by modeling real-world entities as objects.**
- 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 function that operate on it, and protects it from accidental modification from outside function.
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and function around these objects.

- The organization of data and function in object-oriented programs is shown in fig.



- The data of an object can be accessed only by the function associated with that object. However, function of one object can access the function of other objects.
- Some of the features of object oriented programming are:

  a) Emphasis is on data rather than procedure.
  b) Programs are divided into what are known as objects.
  c) Data structures are designed such that they characterize the objects.
  d) Functions that operate on the data of an object are ties together in the data structure.
  e) Data is hidden and cannot be accessed by external function.
  f) Objects may communicate with each other through function.
  g) New data and functions can be easily added whenever necessary.
  h) Follows bottom up approach in program design.

The following table highlights the major differences between Procedural Programming and Object Oriented Programming –

Parameter	Object Oriented Programming	Procedural Programming
Definition	Object-oriented Programming is a programming language that uses classes and objects to create models based on the real world environment. In OOPs, it makes it easy to maintain and modify existing code as new objects are created inheriting characteristics from existing ones.	Procedural Programming is a programming language that follows a step-by-step approach to break down a task into a collection of variables and routines (or subroutines) through a sequence of instructions. Each step is carried out in order in a systematic manner so that a computer can understand what to do.
Approach	In OOPs concept of objects and classes is introduced and hence the program is divided into small chunks called objects which are instances of classes.	In procedural programming, the main program is divided into small parts based on the functions and is treated as separate program for individual smaller program.
Access modifiers	In OOPs access modifiers are introduced namely as Private, Public, and Protected.	No such modifiers are introduced in procedural programming.
Security	Due to abstraction in OOPs data hiding is possible and hence it is more secure than POP.	Procedural programming is less secure as compare to OOPs.
Complexity	OOPs due to modularity in its programs is less complex and hence new data objects can be created easily from existing objects making object-oriented programs easy to modify	There is no simple process to add data in procedural programming, at least not without revising the whole program.
Program division	OOP divides a program into small parts and these parts are referred to as objects.	Procedural programming divides a program into small programs and each small program is referred to as a function.
Importance	OOP gives importance to data rather than functions or procedures.	Procedural programming does not give importance to data. In POP, functions along with sequence of actions are followed.
Inheritance	OOP provides inheritance in three modes i.e. protected, private, and public	Procedural programming does not provide any inheritance.
Examples	C++, C#, Java, Python, etc. are the examples of OOP languages.	C, BASIC, COBOL, Pascal, etc. are the examples POP languages.

BASIC CONCEPTS OF OBJECTS ORIENTED PROGRAMMING

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

- Objects
- Classes
- Data abstraction and encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing

Objects

Definition

- **In object-oriented programming (OOP), an object is an instance of a class. An object represents a specific entity or instance in a program.**
- **It is a runtime entity that encapsulates both data (attributes) and behavior (methods) associated with that particular entity.**

- 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 must handle.
- The fundamental idea behind object oriented approach is to combine both data and function
- Into a single unit and these units are called objects**.**
- An object has a state, which is determined by its attributes or instance variables. These attributes store the object's data and represent its characteristics or properties. For a car object, attributes could include its color, make, model, and current speed.
- In addition to its state, an object can perform actions or behaviors. These behaviors are defined by methods or member functions of the class. Methods are functions associated with the object that allow it to perform specific operations or respond to messages. For example, a car object could have methods such as "start," "accelerate," or "brake."
- Objects are created from a class blueprint using a process called instantiation. When you create an object, you allocate memory for it and initialize its attributes. Each object has its own unique set of attribute values but follows the structure and behavior defined by the class.
- In OOP, objects are the building blocks of the system, and they interact with each other through method calls and message passing.
- Objects can communicate, collaborate, and exchange data to fulfill specific tasks and achieve the overall functionality of the program.
- The concept of objects in OOP enables the modeling of complex systems by breaking them down into modular and manageable entities that have their own state and behavior. This

approach promotes code reuse, encapsulation, and abstraction, allowing for more efficient and maintainable software development.

```
OBJECTS: STUDENT

DATA
        Name
        Date-of-birth
        Marks

FUNCTIONS
        Total
        Average
        Display
        ………
```

Classes

Definition

- **In C++, a class is a user-defined type that serves as a blueprint for creating objects.**
- **It provides a way to define the structure and behavior of objects that belong to a specific class.**
- **A class encapsulates data (member variables) and functions (member functions) that operate on that data.**

- 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 similar types.
- For examples, Mango, Apple and orange members of class fruit.

Create a Class

A class is defined in C++ using **keyword class** followed by the **name of the class**.

The body of the class is defined inside the curly brackets and terminated by a semicolon at the end.

```
class className {
// some data
// some functions
};
```

Syntax to Define Object in C++

```
className objectVariableName;
```

Example 1: Object and Class in C++ Programming

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

// create a class
class Room {

   public:
    double length;
    double breadth;
    double height;

    double calculateArea() {
        return length * breadth;
    }

    double calculateVolume() {
        return length * breadth * height;
    }
};

int main() {

    // create object of Room class
    Room room1;
    // assign values to data members
    room1.length = 42.5;
    room1.breadth = 30.8;
    room1.height = 19.2;

    // calculate and display the area and volume of the room
    cout << "Area of Room =  " << room1.calculateArea() << endl;
    cout << "Volume of Room =  " << room1.calculateVolume() << endl;

    return 0;
}
```

- Here, we defined a class named Room.
- The variables **length**, **breadth**, and **height** declared inside the class are known as **data members.** And, the functions **calculateArea() and calculateVolume() are known as member functions of a class.**
- Room room1; Here Object room1 is created.
- The values of length, breadth, and height are assigned to room1 using dot notation.

- The calculateArea() and calculateVolume() member functions are called on room1 to calculate the area and volume of the room, respectively.
- The calculated area and volume are displayed to the console using cout.

Data Abstraction and Encapsulation

- The wrapping up of data and function into a single unit (called class) is known as **encapsulation**.
- 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.
- In C++, data encapsulation is achieved using access specifiers. The access specifiers define the visibility and accessibility of class members (variables and functions) to the outside world. The three access specifiers in C++ are:

    - **Public**: Public members are accessible from anywhere in the program. They can be accessed by objects of the class and external code that uses the objects. Public members represent the interface of the class, providing a way for external code to interact with the class.

    - **Private**: Private members are only accessible within the class itself. They cannot be accessed directly by objects or external code. Private members are typically accessed and manipulated through public member functions (methods). They represent the internal implementation details and state of the class.

    - **Protected**: Protected members are similar to private members, but they can be accessed by derived classes (classes that inherit from the base class). Protected members provide a way to share data and behaviors between the base class and its derived classes.

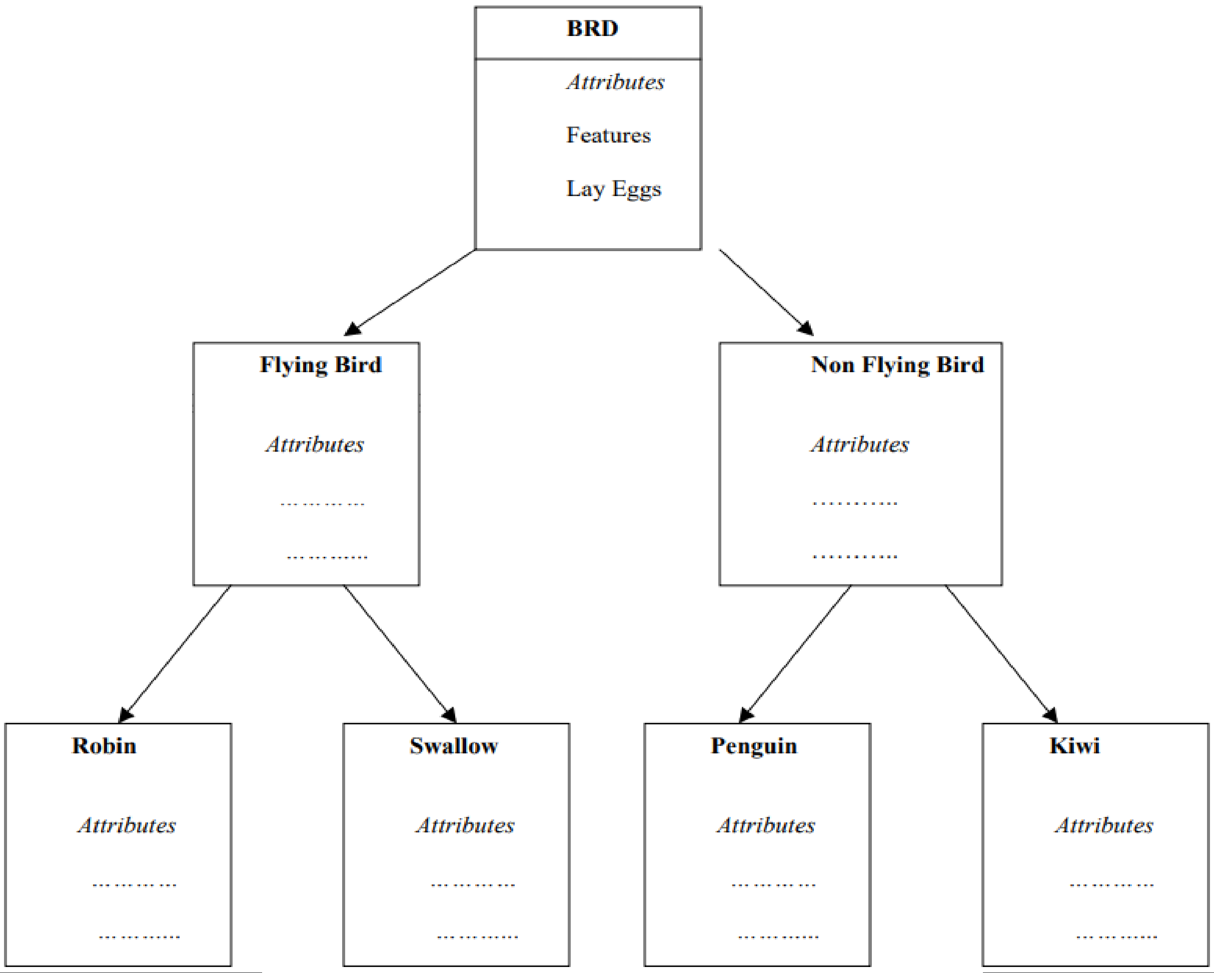
- **Abstraction** refers to the act of representing essential features without including the background details or explanation.
- **Abstraction is the process of only showing the necessary details to the user and hiding the other details in the background**
- Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, wait, and cost, and function operate on these attributes. They encapsulate all the essential properties of the object that are to be created.

INHERITANCE

Inheritance is the process by which objects of one class acquired the properties of objects of another classes.

- This mean that we can add additional features to an existing class without modifying it.
- This is possible by designing a new class will have the combined features of both the classes.
- It enables code reuse, promotes modularity, and establishes hierarchical relationships between classes. Inheritance is a key feature of object-oriented programming, providing a way to model real-world relationships and achieve polymorphism.

Sub Class: The class that inherits properties from another class is called Subclass or Derived Class.
Super Class: The class whose properties are inherited by a subclass is called Base Class or Superclass.
For example, the bird, 'robin' is a part of class 'flying bird' which is again a part of the class 'bird'. The principal behind this sort of division is that each derived class shares common Characteristics with the class from which it is derived as illustrated in fig

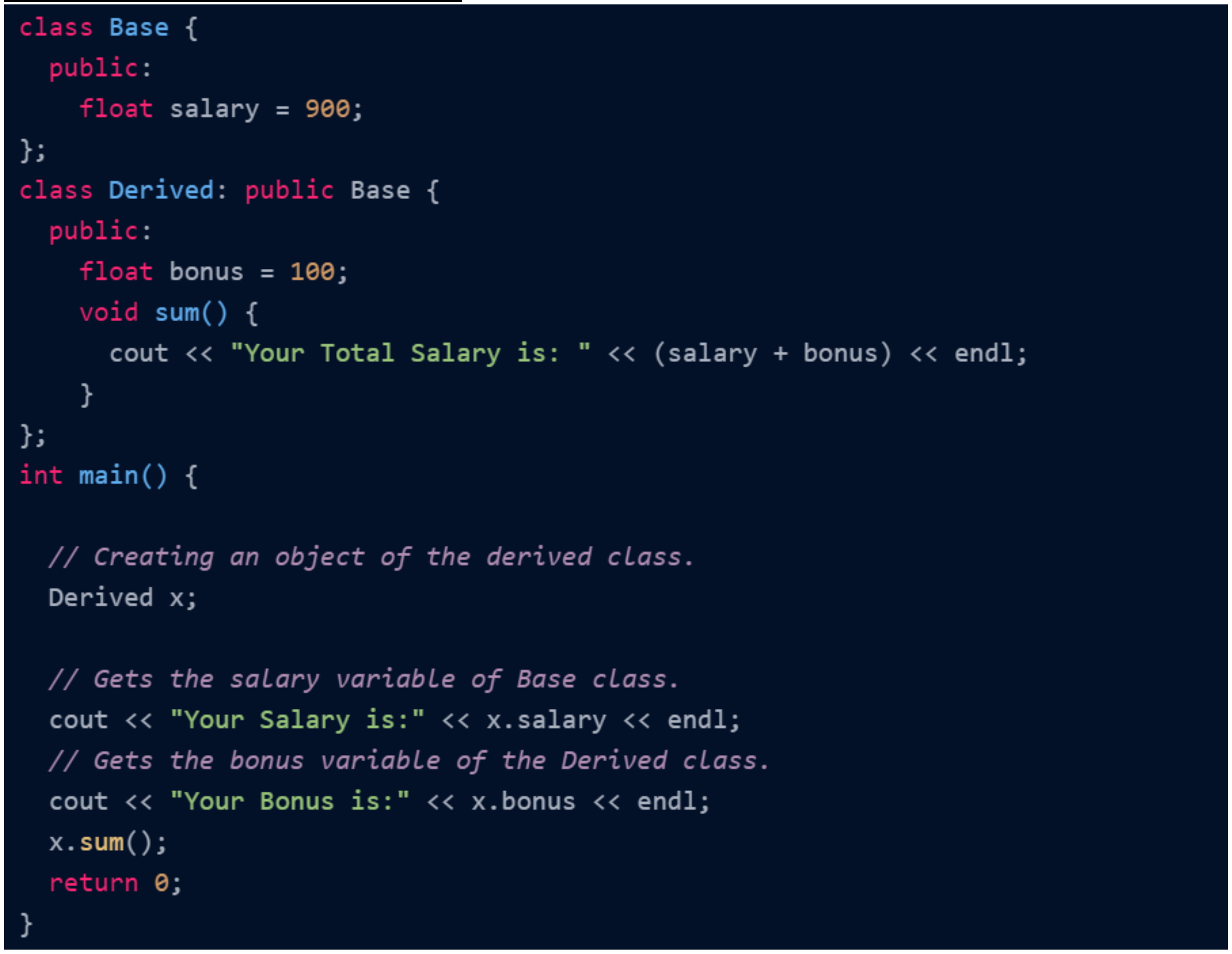


Syntax:

Property Inheritance

```
class  <derived_class_name> : <access-specifier> <base_class_name>
{
 //body
}
```

Where
Class — keyword to create a new class
derived_class_name — name of the new class, which will inherit the base class
Access-specifier — either of private, public or protected. If neither is specified, PRIVATE is taken as default
Base-class-name — name of the base class
Note: A derived class doesn't inherit access to private data members. However, it does inherit a full parent object, which contains any private members which that class declares.
Example Program for Inheritance



```
class Base {
  public:
    float salary = 900;
};
class Derived: public Base {
  public:
    float bonus = 100;
    void sum() {
      cout << "Your Total Salary is: " << (salary + bonus) << endl;
    }
};
int main() {

  // Creating an object of the derived class.
  Derived x;

  // Gets the salary variable of Base class.
  cout << "Your Salary is:" << x.salary << endl;
  // Gets the bonus variable of the Derived class.
  cout << "Your Bonus is:" << x.bonus << endl;
  x.sum();
  return 0;
}
```

In the above example, Base is the class name and the parent class, which contains the property named salary and the value 900.

In the same way, there is another class named Derived, which is the child class, which inherits the property of the parent class and has its property named as a bonus which contains the value of 100.

In the child class, there is a function named sum(), which is used to add the salary and bonus. In the main function, an object is created named "x" of the "Derived" class which is a child class, and using that object, the properties, and the sum function are called from the derived class, which will add the salary and bonus and gives it as output.

POLYMORPHISM

- Polymorphism is another important OOP concept.
- Polymorphism, a Greek term, means the ability to take more than on form. An operation may exhibit different behavior is 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 strings, 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.
- Fig illustrates that a single function name can be used to handle different number and different types of argument. This is something similar to a particular word having several different meanings depending upon the context. Using a single function name to perform different type of task is known as function overloading.

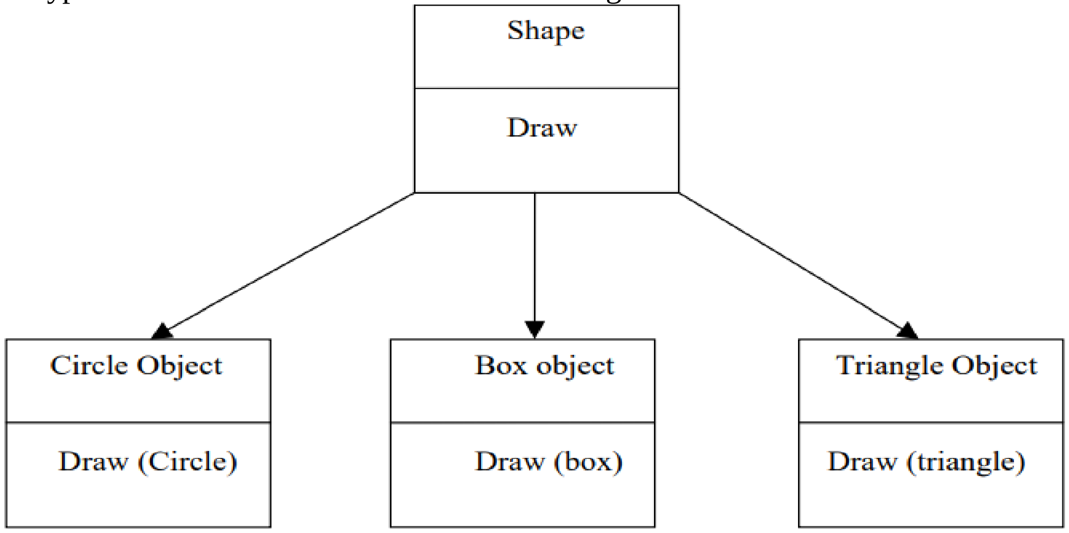


Fig. Polymorphism

- 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 action associated with each operation may differ. Polymorphism is extensively used in implementing inheritance.

DYNAMIC BINDING

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call is not known until the time of the 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.

Dynamic binding in C++ is the mechanism by which the appropriate function implementation is determined at runtime based on the actual object type. It is achieved through the use of virtual functions and the concept of late binding.

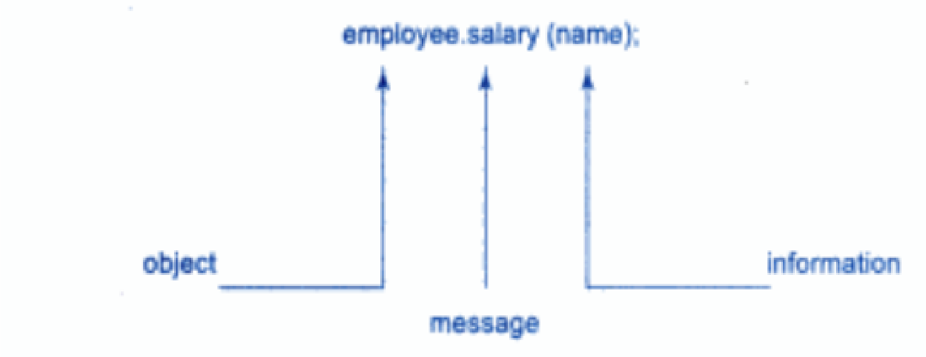
Message Passing

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

1. Creating classes that define object 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 (procedure) in the receiving object that generates the desired results. Message passing involves specifying the name of object, the name of the function (message) and the information to be sent. Example:



Object has 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-oriented contributes tothe solution of many problems associated with the development and quality of software products. The principal advantages are :
1. Through inheritance we can eliminate redundant code and extend the use of existing classes.
2. 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.
3. This principle of data hiding helps the programmer to build secure programs that can't be invaded by code in other parts of the program.
4. It is possible to have multiple instances of an object to co-exist with out any interference.
5. It is easy to partition the work in a project based on objects.
6. Object-oriented systems can be easily upgraded from small to large systems.
7. Message passing techniques for communication between objects makes the interface description with external systems much simpler.
8. Software complexity can be easily managed.

APPLICATION OF OOP:

The most popular application of oops up to now, has been in the area of user interface design such as windows. There are hundreds of windowing systems developed using oop techniques.
Real business systems are often much more complex and contain many more objects with complicated attributes and methods. Oop is useful in this type of applications because it can simplify a complex problem. The promising areas for application of oop includes.
1. Real – Time systems.
2. Simulation and modeling
3. Object oriented databases.
4. Hypertext,hypermedia and expertext.
5. Al and expert systems.
6. Neural networks and parallel programming.
7. Dicision support and office automation systems.
8. CIM / CAM / CAD system.