

OBJECT ORIENTED PROGRAMMING USING C++

Prof. Priya.V.A.P(Sr), SITE, VTU

Unit No. 1

OBJECT ORIENTED PARADIGM

- o Evolution of Object Oriented Programming (OOP) Paradigm- Concepts of OOP – Data Abstraction – Encapsulation – Class – Inheritance – Polymorphism – I/O Streams – Merits and Demerits of Object Oriented Paradigm.

Paradigm Shift in Programming

- Programs: Data, Statements, Functions
- Programming with flowchart
- Program = Data Structure + Algorithms
- Structured Programming
 - Sequence Instructions
 - Decision Making
 - Looping

ALGORITHMS

- An algorithm is a sequence of precise instructions for solving a problem in a finite amount of time.







Properties of an Algorithm:

- It must be precise and unambiguous (accurate meaning)
- It must give the correct solution in all cases
- It must eventually end.

Understanding the Algorithm

- Possibly the simplest and easiest method to understand the steps in an algorithm, is by using the **flowchart** method. This algorithm is composed of block symbols to represent each step in the solution process as well as the directed paths of each step. (Pictorial Representation of an Algorithm)

The most common block symbols are:

Symbol	Representation	Symbol	Representation
	Start/Stop		Decision
	Process		Connector
	Input/Output		Flow Direction

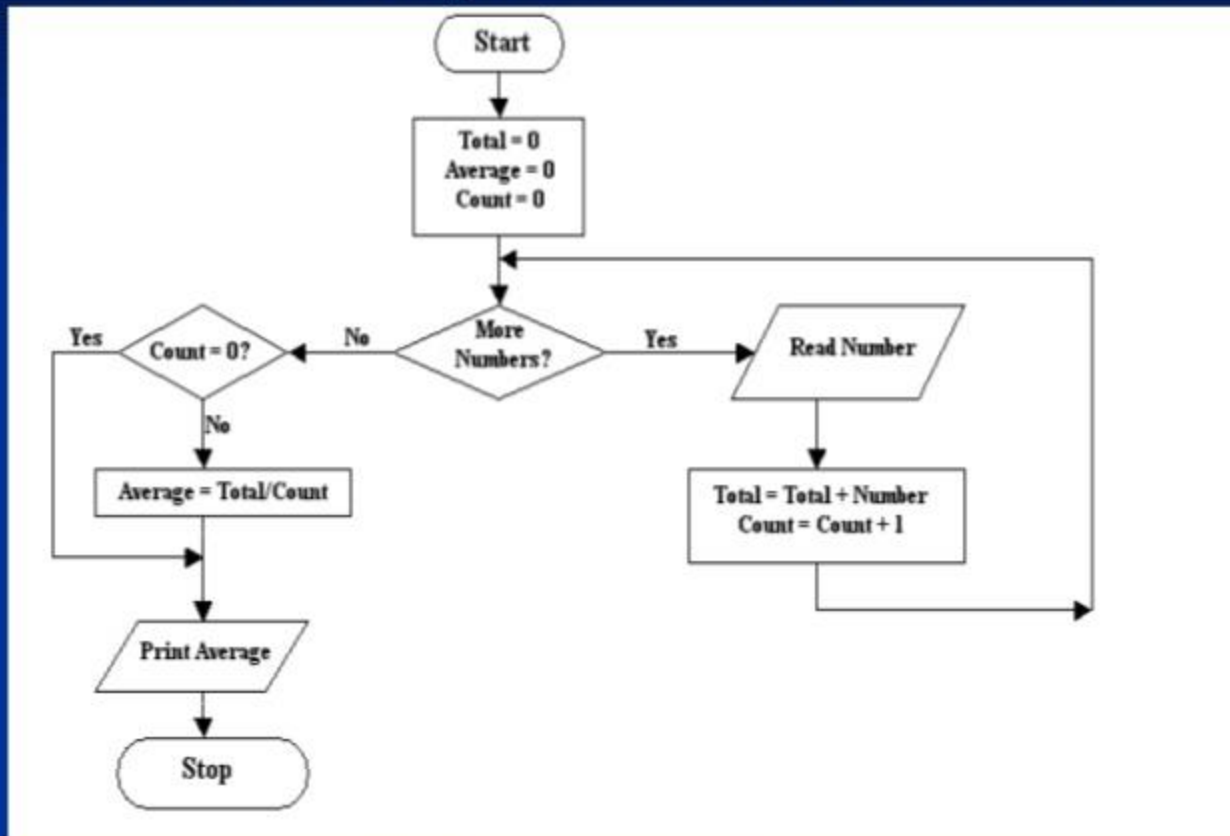
Problem for Flowchart

- A variable is a symbolic name assigned to a computer memory which stores a particular value.

e.g. COUNTER, SUM, AVE

Calculate its Average ?

A flowchart representation of the algorithm for the above problem can be as follows:



History of Object-Oriented Programming

- **SIMULA I** (1962-65) and **SIMULA 67** (1967) were the first two object-oriented languages.
- Developed at the Norwegian Computing Center, Oslo, Norway by Ole-Johan Dahl and Kristen Nygaard .
- Simula 67 introduced most of the key concepts of object-oriented programming: objects and classes, subclasses (“inheritance”), virtual procedures.

Object-Oriented Programming

- Programming for simulation
- Software Crisis
- Software Reuse
- Software IC
- Polymorphism, Inheritance and Encapsulation (PIE).
- Classes as an Abstract Data Type (Abstraction)
- Easy to debug and maintain
- Mainstream in software development
- Software components.

Object Oriented Languages

- Eiffel (B. Meyer)
- CLOS (D. Bobrow, G. Kiczales)
- SELF (D. Ungar et al.)
- Java (J. Gosling et al.)
- BETA (B. Bruun-Kristensen, O. Lehrmann Madsen, B. Møller-Pedersen, K. Nygaard)
- Other languages add object dialects, such as TurboPascal
- C++ (Bjarne Stroustrup)

Structured Programming Concept(Modular Prog.)

- Structured programming techniques assist the programmer in writing effective error free programs.
- Top down approach
- Overall program structure divided into separate subsections
- This technique help to make isolated small pieces of code easier to understand without having to understand the whole program at once.

Cont....

- After a piece has been tested and studied in detail individually, it is then integrated into the overall program structure.
- It is possible to write any computer program by using only three (3) basic control structures (logical concept's) :
 - Sequence instructions
 - Decision Structure(if-else)
 - Loop (While, Do While,...)
- Ex. are Pascal , C, ADA.

Procedural Programming (Procedure oriented)

- Top down approach
- Procedures, also known as functions or methods simply contains a series of computational(Algorithmic) steps to be carried out.
- procedural programming specify the syntax and procedure to write a program.
- Big program is a divided into small pieces.
- Functions are more important than data.
- Input- arguments, output-return values.
- Ex. are C, Algol etc.

Top down approach

- A complex program divides into smaller pieces, makes efficient and easy to understand a program.
- Begins from the top level.
- Emphasize the planning and complete understanding of a program
- No coding can begins until a sufficient level of module details has been reached.

Advantages of the Top-Down Design Method

- **It is easier to comprehend the solution of a smaller and less complicated problem than to grasp the solution of a large and complex problem.**
- **It is easier to test segments of solutions, rather than the entire solution at once. This method allows one to test the solution of each sub-problem separately until the entire solution has been tested.**
- **It is often possible to simplify the logical steps of each sub-problem, so that when taken as a whole, the entire solution has less complex logic and hence easier to develop.**
- **A simplified solution takes less time to develop and will be more readable.**
- **The program will be easier to maintain.**

Bottom up approach

- Reverse top down approach.
- Lower level tasks are first carried out and are then integrated to provide the solution of a single program.
- Lower level structures of the program are evolved first then higher level structures are created.
- It promotes code reuse.
- It may allow unit testing.

Programming Languages

- Programming languages allow programmers to code software.
- The three major families of languages are:
 - Machine languages
 - Assembly languages
 - High-Level languages

Machine Languages

- Comprised of 1s and 0s
- The “native” language of a computer
- Difficult to program – one misplaced 1 or 0 will cause the program to fail.
- Example of code:

1110100010101

10111010110100

111010101110

10100011110111

Assembly Languages

- Assembly languages are a step towards easier programming.
- Assembly languages are comprised of a set of elemental commands which are tied to a specific processor.
- Assembly language code needs to be translated to machine language before the computer processes it.
- Example:
ADD 1001010, 1011010

High-Level Languages

- High-level languages represent a giant leap towards easier programming.
- The syntax of HL languages is similar to English.
- Historically, we divide HL languages into two groups:
 - Procedural languages
 - Object-Oriented languages (OOP)

Procedural Languages

- Early high-level languages are typically called procedural languages.
- Procedural languages are characterized by sequential sets of linear commands.
- The focus of such languages is on *structure*.
- Examples include C, COBOL, Fortran, LISP, Perl, HTML, VBScript

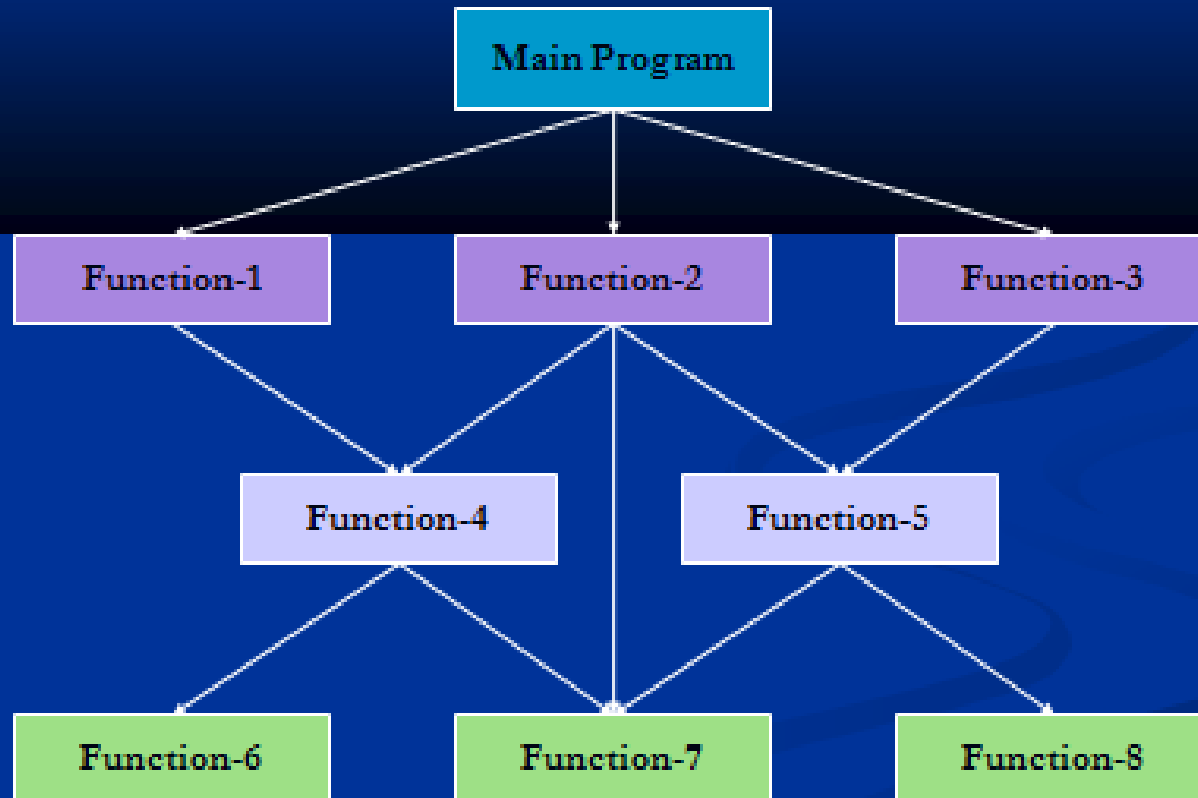
Object-Oriented Languages

- Most object-oriented languages are high-level languages.
- The focus of OOP languages is not on structure, but on *modeling data*.
- Programmers code using “blueprints” of data models called *classes*.
- Examples of OOP languages include C++, Visual Basic.NET and Java.

Procedure-Oriented Programming

- Conventional programming using high level languages like COBOL, FORTRAN, C, etc.
- The problem is viewed as a sequence of things to be done.
- The primary focus is on functions.
- Procedure-oriented programming basically consists of writing a list of instructions for the computer to follow and organizing these instructions into groups known as functions.

Typical structure of procedure-oriented program

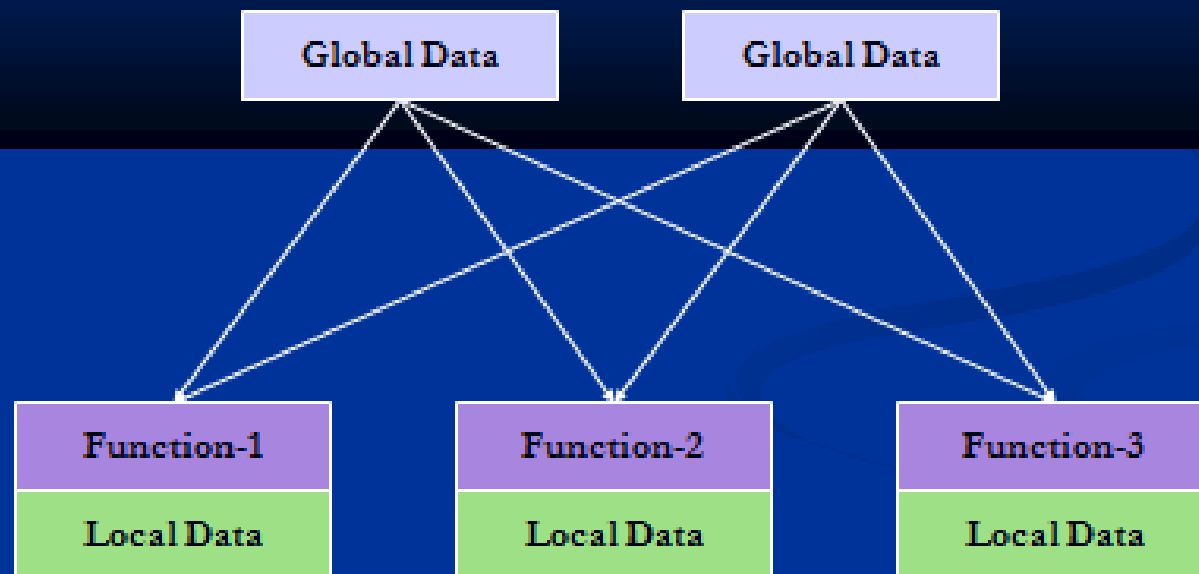


Procedure-Oriented Programming

continue ...

- To revise an external data structure, we also need to revise all functions that access the data.
- This approach does not model real world problems.
- This is because functions are action-oriented and do not really correspond to the elements of the problem.

Relationship of data and functions in procedural programming



Characteristics of Procedure-Oriented Programming

- Emphasis is on doing things.
- 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 Programming

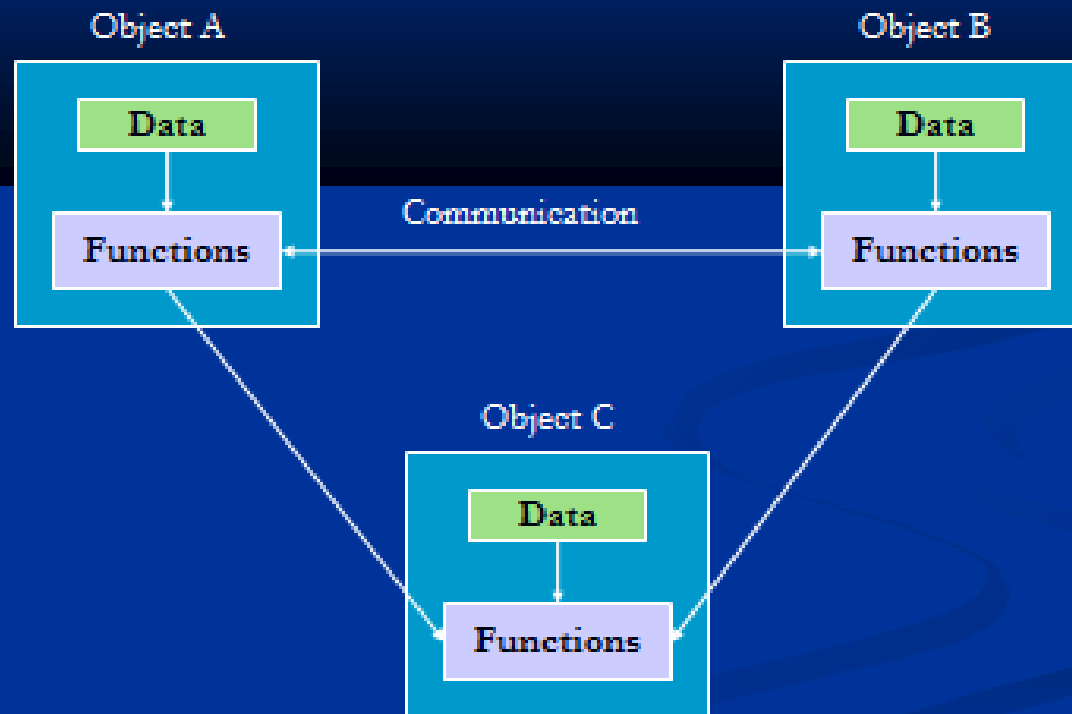
- OOP treat data as a critical element in the program development and does not allow it to flow freely around the system.
- It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions.
- OOP allows decomposition of a problem into a number of entities called objects and then build data functions around these objects.

Object-Oriented Programming

continue ...

- The data of an object can be accessed only by the functions associated with that object.
- Functions of one object can access the functions of another objects.

Organization of data and functions in OOP



Characteristics of Object-Oriented Programming

- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together in the data structure.
- Data is hidden and can not be accessed by external functions.

Characteristics of Object-Oriented Programming

continue ...

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

Object-Oriented Programming

■ Definition:

It is 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 the object is considered to be a partitioned area of computer memory that stores data and set of operations that can access that data.

Basic Concepts of Object-Oriented Programming

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

Basic Concepts of OOP

continue ...

■ Objects

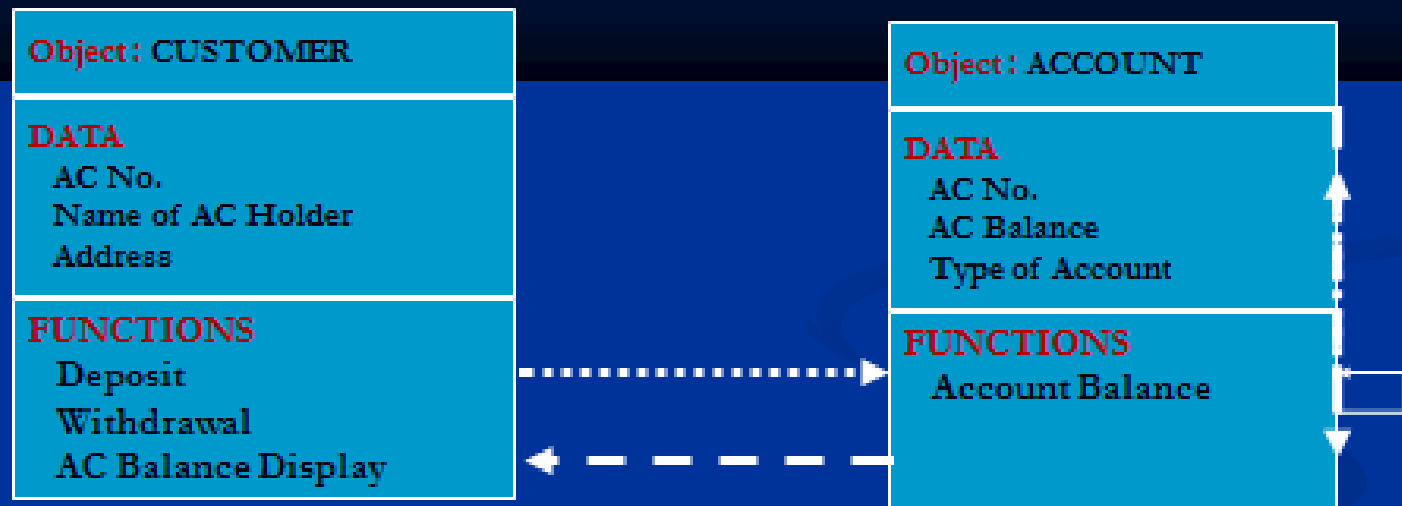
- Objects are the basic run-time entities in an object-oriented system.
- They may represent a person, a place, a bank account, etc.
- Objects take up space in the memory and have an associated address like a structure in C.

When a program is executed, the objects interact by sending messages to one another.

Basic Concepts of OOP

continue ...

■ Objects



Basic Concepts of OOP

■ Classes

continue ...

Classes are user-defined data types.

- The entire set of data and code of an object can be made a user-defined data type with the help of a class.
- 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 a collection of objects of similar type.

Basic Concepts of OOP

continue ...

■ Classes

If `fruit` has been defined as a class, then the statement

`fruit mango;`

will create an object `mango` belonging to the class `fruit`.

Concept of Class and Object

- “**Class**” refers to a blueprint. It defines the variables and methods the objects support. It is the basic unit of Encapsulation. It also defines as the Collection of a similar types of objects.
- “**Object**” is an instance(Properties) of a class. Each object has a class which defines its data and behavior.

Structure of a Class in C++

```
class name {
```

declarations

← attributes and
symbolic constants

constructor definition(s)

← how to create and
initialize objects

method definitions

← how to manipulate
the state of objects

```
}
```

These parts of a class can
actually be in any order

Sample class

```
#include<iostream.h>
class Pencil
{
    public String color = "red";
    public int length;
    public float diameter;
        setcolor(string);

    public void setColor (String
newColor) {
        color = yellow;
    }
}
```

Members of class

private: private members are accessible

only in the class itself.

protected: protected members are accessible in classes in the same package, in subclasses of the class and inside the class.

public: public members are accessible anywhere (outside the class).

Features of OOP

- Polymorphism
- Inheritance
- Encapsulation
(PIE)

Basic Concepts of OOP

continue ...

■ Polymorphism

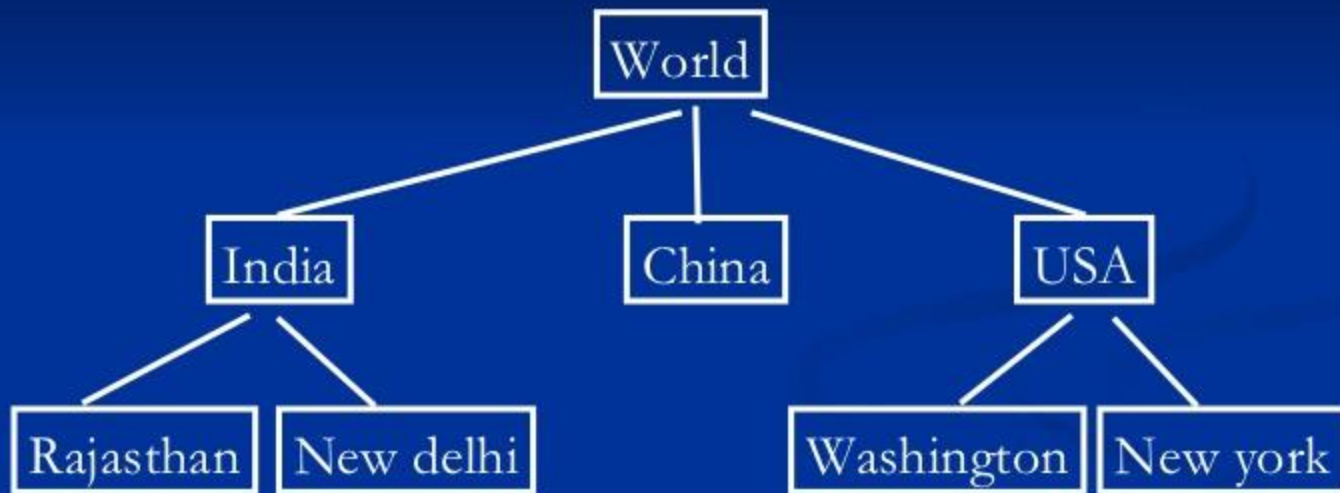
- ability to take more than one form

- An operation may exhibit different behaviours in different instances.
- The behaviour depends upon the types of data used in the operation.

Polymorphism

- “Poly”= Many, “Morphism”= forms
- For ex. We want to find out max. out of three no., We can pass integer, float etc.
- Two types –
 - Compile time polymorphism.
 - Run time polymorphism.

Polymorphism



Basic Concepts of OOP

continue ...

■ Inheritance

- Inheritance provides the idea of re usability
- We can add additional features to an existing class without modifying it.
(By deriving new class from existing one. The new class will have the combined features of both the classes.)

Inheritance

- Mechanism of deriving a new class from an already existing class.
- 5 types of inheritance
 - Single level
 - Multilevel
 - Multiple
 - Hierarchical
 - Hybrid

Flower



Rose



Base class



Derived class

Single level

Parrot

Sparrow



Base class



Bird



Derived class

Multiple

World



India



Rajasthan



Jaipur

Multi level


```
Class base
{
Data members and
Functions;
};
Class
derived:public
base
{
Data members and
functions;
};
```

Single Level

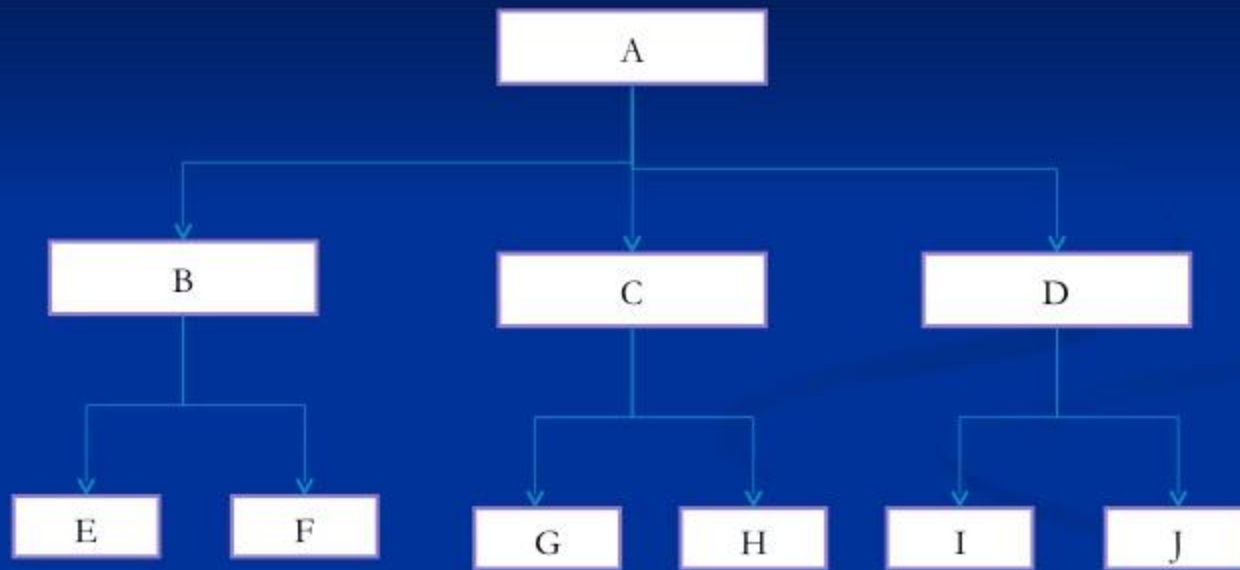
```
Class A
{
};
Class B:public A
{
};
Class C:public B
{
};
Class D: public C
{
};
```

Multi level

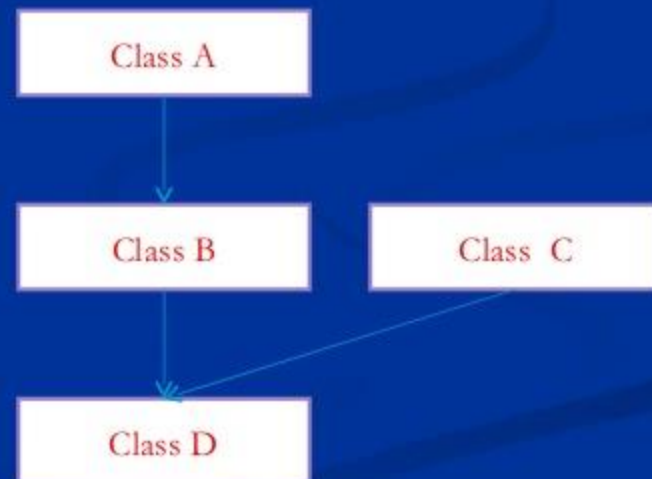
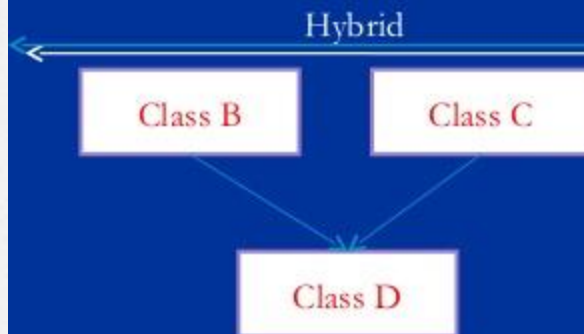
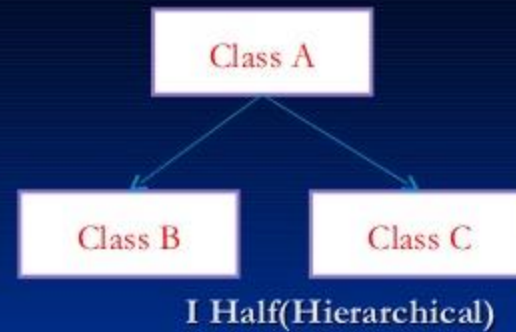
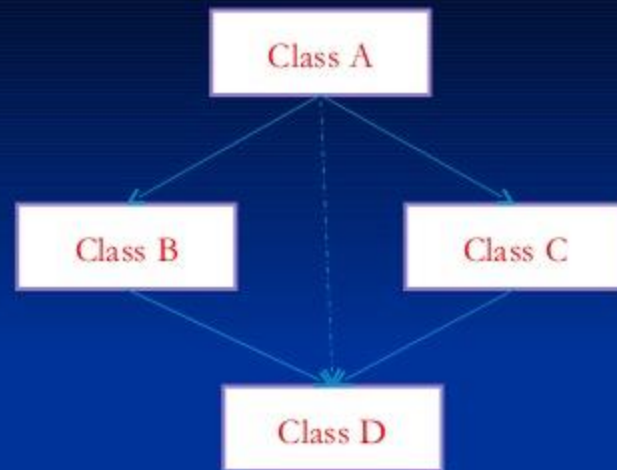
```
Class A
{
};
Class B
{
};
Class C:public A, Public
B
{
};
```

Multiple

Hierarchical Inheritance

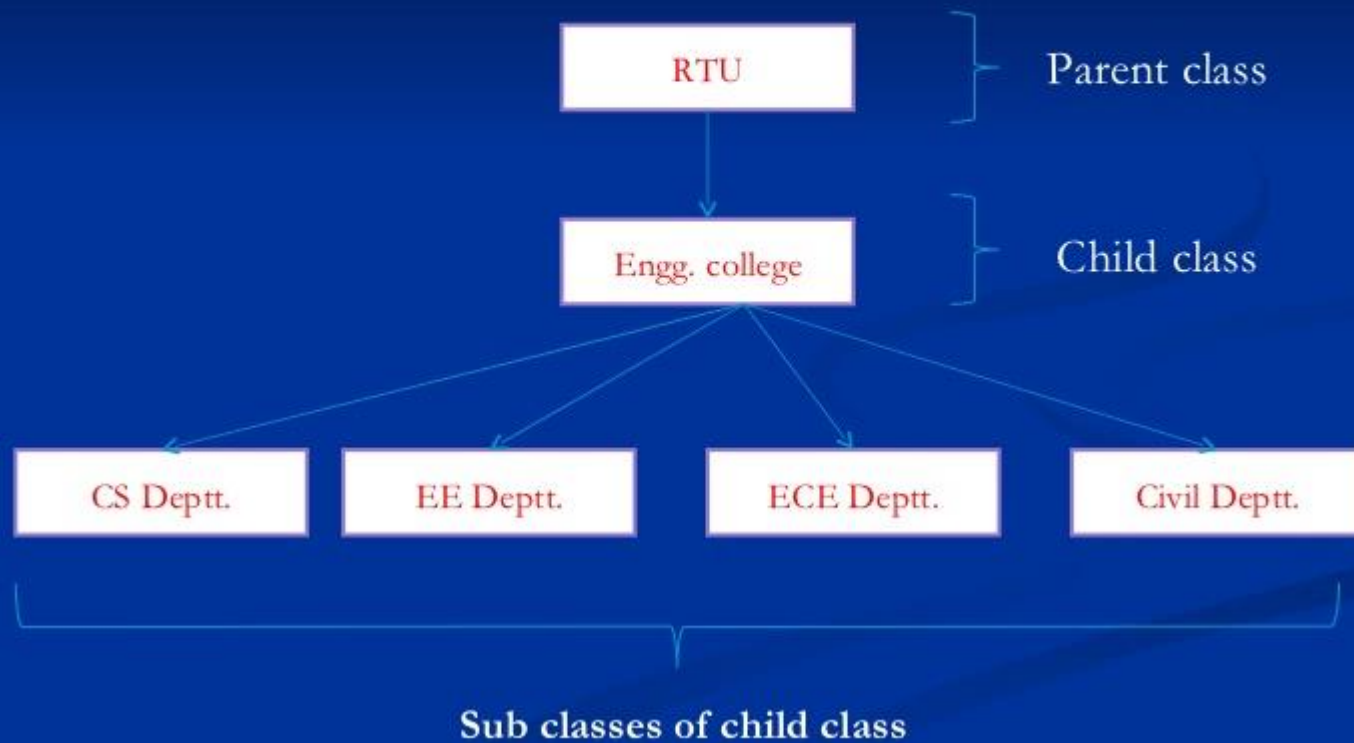


Hybrid = Hierarchical + Multiple



Hybrid = Multi level+ Multiple

Ex. of Inheritance



PPP INHERITANCE (CLASS MEMBERS)

➤ PUBLIC :

Class B: public A

```
{  
};
```

** the line class B: public A tells the compiler that we are inheriting class A in class B in public mode. In public mode inheritance note the followings:

- f. all the public members of class A becomes public members of class b
- g. All the protected members of class A becomes protected members of class B
- h. Private members are never inherited.

Basic Concepts of OOP

continue ...

■ Data Abstraction and Encapsulation

- The wrapping up of data and functions into a single unit 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.

Basic Concepts of OOP

continue ...

■ Data Abstraction and Encapsulation

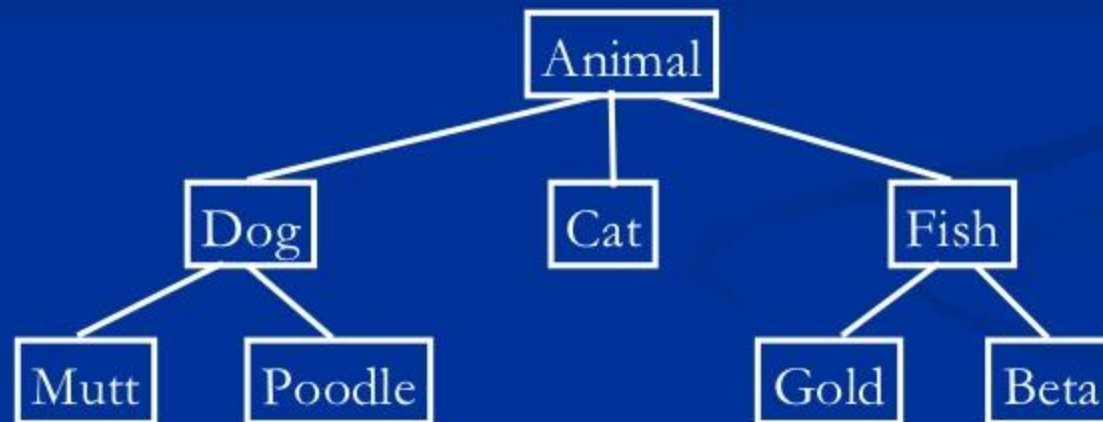
The attributes wrapped in the classes are called data members and the functions that operate on these data are called methods or member functions.

Since the classes use the concept of data abstraction, they are known as Abstracted Data Types (ADT).

Encapsulation

Encapsulation is the mechanism that binds the data & function in one form known as class. The data & function may be private or public.

Objects binds together in form of a Class...



Basic Concepts of OOP

continue ...

■ Dynamic Binding

Binding refers to the linking of a procedure call to the code to be executed in response to the call.

Dynamic binding (late 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.

Basic Concepts of OOP

continue ...

■ Message Passing

- An oop consists of a set of objects that communicate with each other.
- OOP involves the following steps:
 - Creating classes that define objects and their behaviour.
 - Creating objects from class definitions.
 - Establishing communication among objects.
- Objects communicate with one another by sending and receiving information.

Basic Concepts of OOP

continue ...

■ Message Passing

- A message for an object is a request for execution of a procedure.
- The receiving object will invoke a function and generates results.
- Message passing involves specifying:
 - The name of the Object.
 - The name of the Function.
 - The information to be send.

Benefits of OOP

- Inheritance – eliminate redundant code and extend the use of existing classes.
- We can build programs from the standard working module, no need of starting from the scratch.
- Data hiding helps the programmer to build secure programs that can not be invaded by code in other parts of the program.

Benefits of OOP

continue ...

- Multiple instances of an objects can co-exists with out any interference.
- It is easy to partition the work in a project based on objects.
- Object-oriented system 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.