# CHAPTER I

## OBJECT ORIENTED CONCEPTS USING C++

### 1.1 Object Oriented Programming Language

A computer is a tool to solve a wide range of problems. The solutions to the problems are in the form of computer programs or application software. These programs are written using a chosen programming language.

A computer program operates on a set of known input data items. The program transforms this input data into a set of expected data items. Only this set of expected data items must be the output of the computer program.

In the early programming languages the input and output data items were represented as variables. Data types categorized these input data items. Control statements provided a way of instructing the computer on the operations that need to be performed on the data items.

Programming languages have another use. They help us in organizing our ideas about the solution of the problem. As the problems being solved or the applications being developed became complex, this aspect of programming languages became very important. Many programming languages emerged to address this issue along with the ease of instructing the computer.

It was realized that viewing the solution of a problem as two separate segments ‘data’ and ‘operations’ does not resemble the way human beings solve the real life problems.

Object oriented programming languages such as C++ are based on the way human beings normally deal with the complex aspects

of real life. It has been observed that human beings normally solve real life problems by identifying distinct objects needed for the solution. Human beings then recognize the relationships amongst these objects. The relationships are like 'part of the whole' or are 'a type of'. Simple abilities such as recognizing that one object is a part of the bigger object and one object is a type of another object are proving to be very important in solving problems in real life. Object Oriented programming facilitates this way of problem solving by combining 'data' and 'operations' that are to be performed on the data.

In other words, the set of data items is split into smaller groups such that a set of operations can be performed on this group without calling any other function. This group of data and the operations together are termed - 'object'. The operations represent the behavior of the object. An object attempts to capture a real world object in a program.

For example, take a look at any calculator, it has both state and behaviour. Its state refers to its physical features like its dimensions, buttons, display screen, operators and the like. Its behaviour refers to the kind of functions it can perform like addition, subtraction, storing in memory, erasing memory and the like.

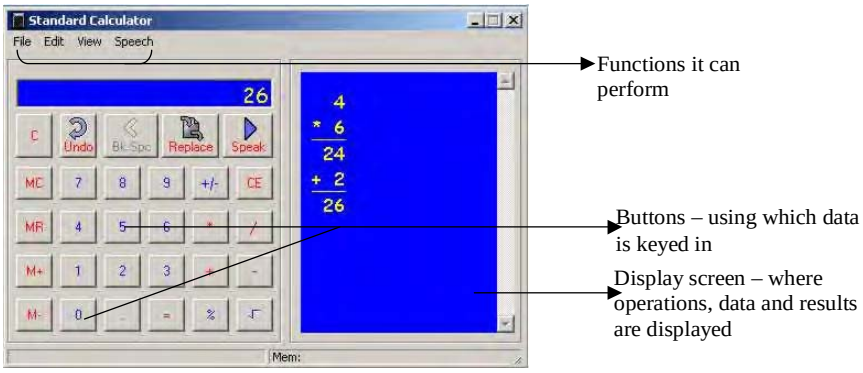



**Fig. 1.1** **Standard Calculator**

In an object oriented programming language, a calculator is viewed as follows :

**Object – calculator**
**Data :**
Number1,result, operator, Number_backup

**Functions :**
Additon()
Subtraction()
Erase_Memory()
Display_Result()

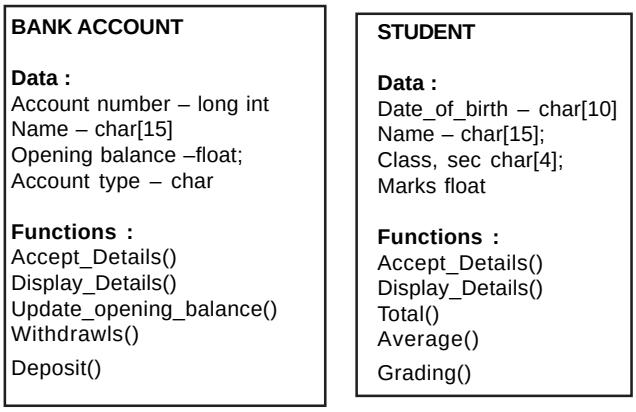
- ✓ **An object is a group of related functions and data that serves those functions.**
- ✓ **An object is a kind of a self-sufficient "subprogram" with a specific functional area.**

The process of grouping data and its related functions into units called as objects paves way for **encapsulation**.

**The mechanism by which the data and functions are bound together within an object definition is called as ENCAPSULATION.**

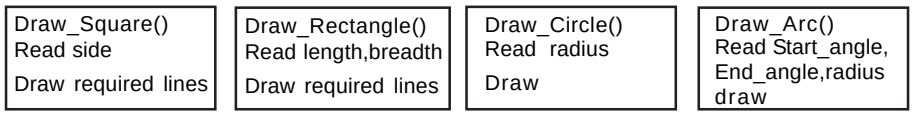
It is easy to see how a bank-account, a student, a bird, a car , a chair etc., embodies both state and behaviour. It is this resemblance to real things that gives objects much of their power and appeal. Objects make it easy to represent real systems in software programs.

Examples of objects – BANK ACCOUNT & STUDENT




## 1.2 Polymorphism

Now let us consider the job of drawing different shapes like a rectangle, square, circle and an arc. We tend to define different functions to draw these different shapes. The definitions may be like this :




Now look at the following function :

Draw( side) – is defined to draw a square
Draw (length, breadth) - is defined to draw a rectangle
Draw(radius) - is defined to draw a circle
Draw(radius,start_angle,end_angle) – to draw an arc

The function draw() accepts different inputs and performs different functions accordingly. As far as the user is concerned, he will use the function **draw()** to draw different objects with different inputs. This differential response of the function draw() based on different inputs is what is called as **polymorphism**.

The ability of an object to respond differently to different messages is called as **polymorphism**.

## 1.3 Inheritance

The data type **Class** conventionally represents an object in the real world. Class is a template for entities that have common behaviour. For example animals form a group of living beings, or in other words **animals** is a class. We know that animals are divided into mammals, reptiles, amphibians, insects, birds and so on. All animals share common behaviour and common attributes. Eyes, skin, habitat, food refer to the features or attributes of the animals, while reproduction, living_style, prey_style etc refers to the behaviour of the animals. Every sub group of animals has its own unique features or styles apart from the common behaviour and features. The sub groups do share the properties of the parent class – "ANIMALS" apart from its own sub classes viz ., mammals, reptiles, amphibians, insects, birds. This may be pictorially depicted as follows :

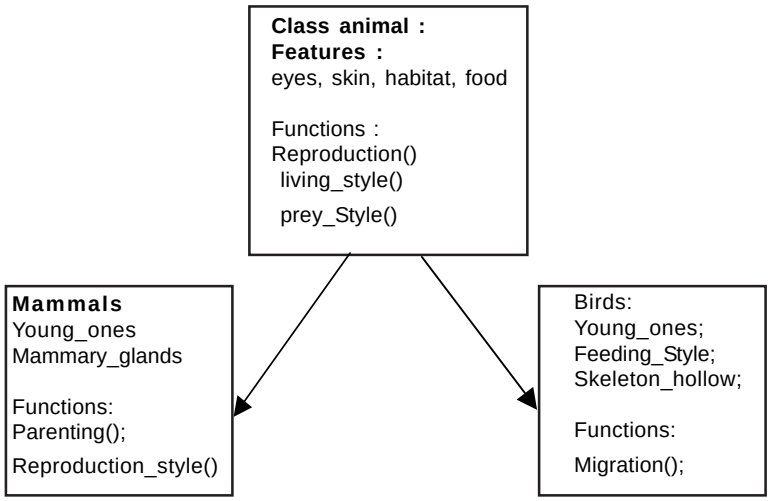



**Fig. 1.1 Inheritance**

Animals is called the base class, and Mammals and Birds are called derived classes. The derived classes are power packed, as they include the functionality of the base class along with their own

unique features. This process of acquiring the Base class properties is what is called **inheritance** .

Inheritance increases the functionality of a derived class and also promotes reusability of code (of the base class).

Advantages of Object Oriented Programming –

- ✓ Class data type allows programs to organize as objects that contain both data and functions .
- ✓ Data hiding or Abstraction of data provides security to data, as unrelated member functions(functions defined outside the class) cannot access its data, or rather it reveals only the essential features of an object while curtailing the access of data
- ✓ Polymorphism reduces software complexity, as multiple definitions are permitted to an operator or function
- ✓ Inheritance allows a class to be derived from an existing class , thus promoting reusability of code, and also promote insertion of updated modules to meet the requirements of the dynamic world

## 1.4 A Practical Example : Domestic Waterusage

**Fig.1.2 Domestic Waterusage**

For example, let us consider developing a program that modeled home water usage. The objective of this program is to compute the water consumed by each outlet in a building and also total consumption. All that we require for this program is the number of taps installed in the building, amount of water that flowed through each tap, and finally the amount of water consumed. Each tap may be viewed as an object. The functions associated would be to start and stop the flow of water, return the amount of water consumed in a given period, and so on. To do this work, the tap object would need instance variables to keep track of whether the tap is open or shut, how much water is being used, and where the water is coming from. The object may be visualised as:

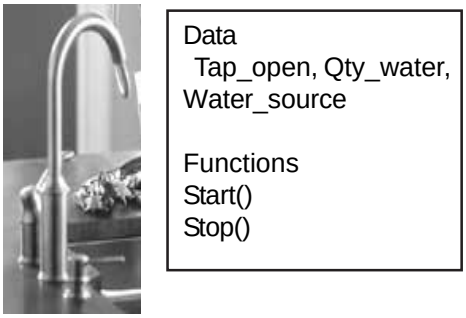



**Fig.1.3 Tap as an Object**

The program that models water usage will also have WaterPipe objects that delivers water to the taps . There could be a Building object to coordinate a set of WaterPipes and taps. When a Building object is asked as to how much water is being used, it might call upon each tap and pipe to report its current state. The project may be visualised as shown in Fig.1.4.

Now the total_amount of water consumed would be calculated as t1.water_consumed() + t2.water_consumed+t3.water_consumed() and water consumption by each outlet would be given away individually by t1.water_consumed, t2.water_consumed() and so on.

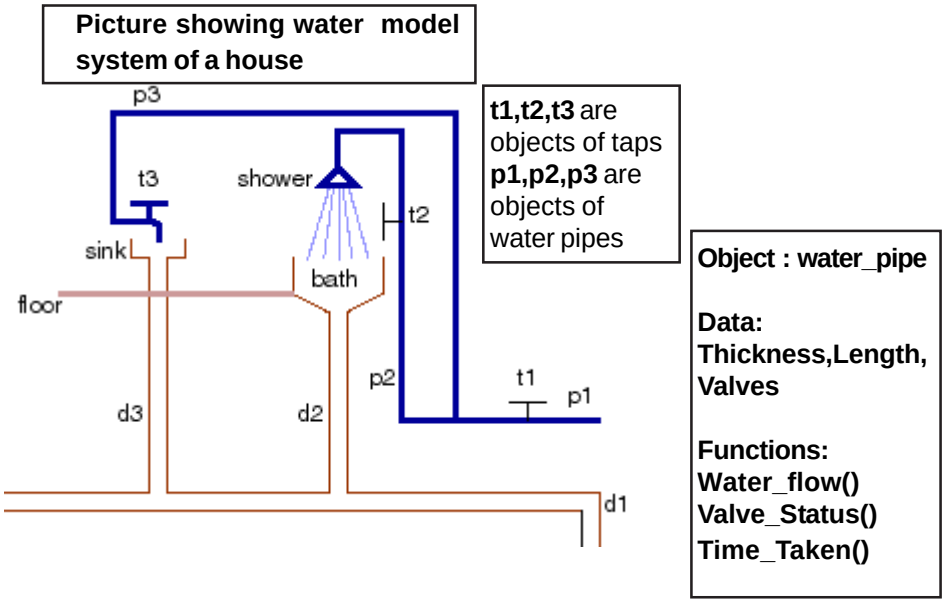



**Fig.1.4 Water Distribution System in a House**

t1.water_consumed() would in turn communicate with p1 to get the amount of water flowed through that pipe, as tap-1s(t1) water consumption is determined by pipe1(p1). Thus a program consists of objects that call each other to compute. Each object has a specific role to play, and the co-ordinated working of all the modules produces the end result of a program. Objects communicate with one another by sending data as inputs.

Everyday programming terminology is filled with analogies to real-world objects like tables, students, managers, bank accounts, and the like. Using such entities as programming objects merely extends the comparison in a natural way. This line of thinking about functions and object behaviour is the key factor of object-oriented programming.

## Exercises

### I. Fill in the blanks

a. _____ model entities in the real world

b. Binding of data and member functions together is called as _____

c. The ability of an object to respond differently to different messages is called as _____

d. The process of creating new data types from existing data type is called as _____

### II. Answer the following briefly

1. What is the significance of an object ?
2. What is Encapsulation?
3. How is polymorphism different from inheritance?

### III Design data type for the following project

A company wishes to prepare a data model for its activities. The company stores information of all its employees. The common details of all employees are : Name, date_of_birth,language and nativity.

Additional details of employees based on their placement are stored as :

a. Stores – date of joining, dept, salary

b. Scientist – area of specialisation, current project details, paper_presentations

c. Technician – Height, Weight, ailments, risk factor, department, wages.