UNIT 1 INTRODUCTION & FEATURES

Procedure-Oriented Programming

In the procedure oriented approach, the problem is viewed as the sequence of things to be done such as reading, calculating and printing such as Cobol, Fortran and c. The primary focus is on functions. A typical structure for procedural programming is shown in fig.1.2. The technique of hierarchical decomposition has been used to specify the tasks to be completed for solving a problem.

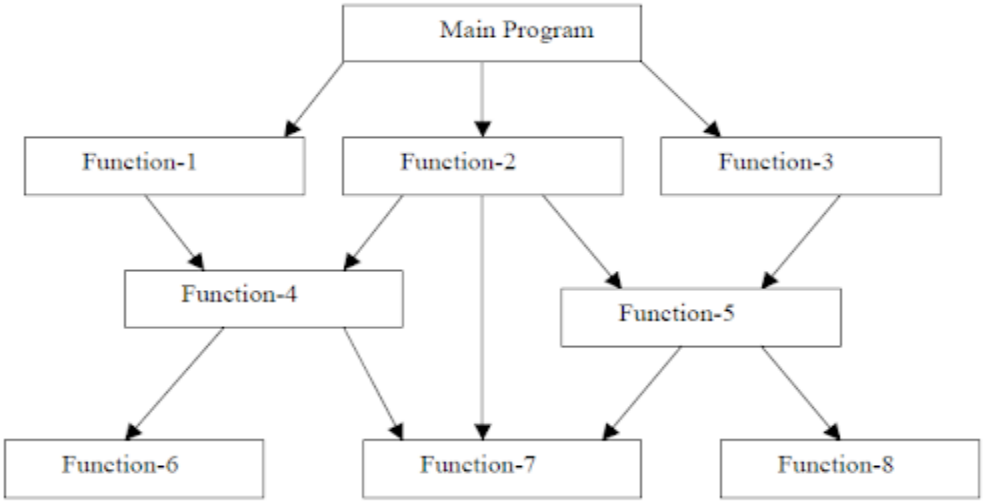


Fig. 1.2 Typical structure of procedural oriented programs

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.

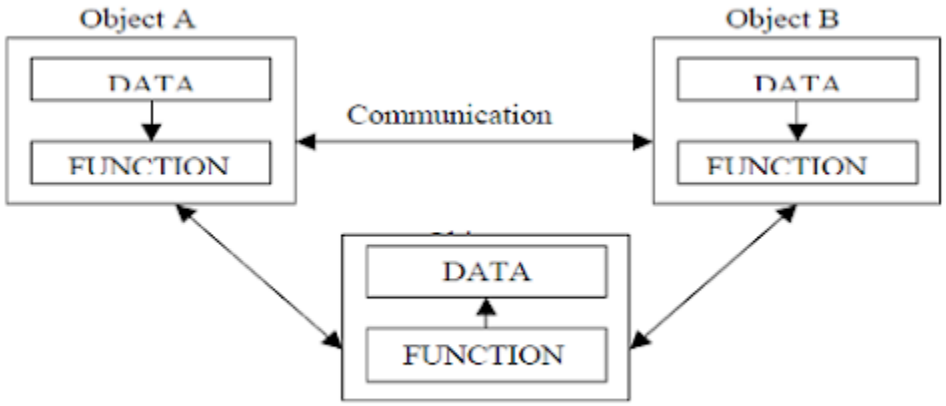
Another serious drawback with the procedural approach is that we do not model real world problems very well. This is because functions are action-oriented and do not really corresponding to the element of the problem.

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

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.1.3. 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.



Organization of data and function in OOP

Some of the features of object oriented programming are:

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

Difference Between OOP and POP

Both Procedural Oriented Programming (POP) and Object Oriented Programming (OOP) are the high level languages in programming world and are widely used in development of applications. On the basis of nature of developing the code both languages have different approaches on basis of which both are differentiate from each other.

Following are the important differences between Procedural Oriented Programming (POP) and Object Oriented Programming (OOP)

Sr. No.	Key	Object Oriented Programming (OOP)	Procedural Oriented Programming (POP)
1	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.	On other hand Procedural Oriented 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.
2	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.	On other hand in case of POP the the main program is divided into small parts based on the functions and is treated as separate program for individual smaller program.
3	Access modifiers	In OOPs access modifiers are introduced namely as Private, public, and Protected.	On other hand no such modifiers are introduced in POP.
4	Security	Due to abstraction in OOPs data hiding is possible and hence it is more secure than POP.	On other hand POP is less secure as compare to OOPs.

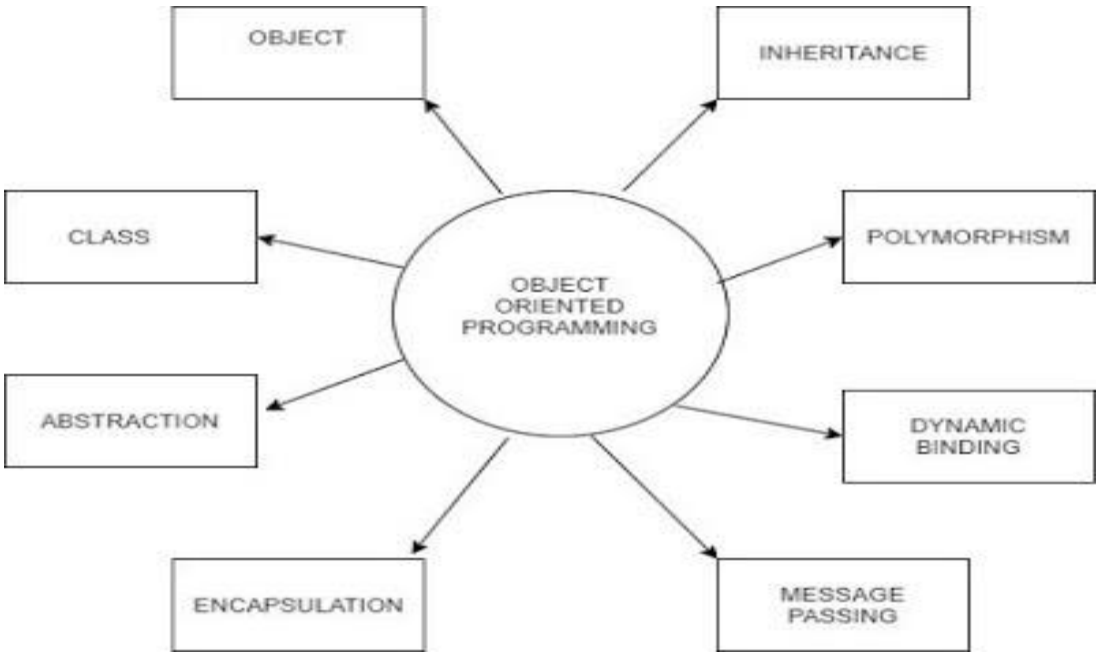
Sr. No.	Key	Object Oriented Programming (OOP)	Procedural Oriented Programming (POP)
5	Complexity	OOPs due to modularity in its programs is less complex and hence new data objects can be created easily from existing objects making object-oriented programs easy to modify	On other hand there no simple process to add data in POP at least not without revising the whole program.
6	Example	C++, JAVA, VB.NET, C#.NET.	C, VB, FORTRAN, Pascal

Difference between C and C++

C	C++
C was developed by Dennis Ritchie between the year 1969 and 1973 at AT&T Bell Labs.	C++ was developed by Bjarne Stroustrup in 1979.
C does no support polymorphism, encapsulation, and inheritance which means that C does not support object oriented programming.	C++ supports polymorphism, encapsulation, and inheritance because it is an object oriented programming language.
C is a subset of C++.	C++ is a superset of C.
C contains 32 keywords.	C++ contains 52 keywords.
C supports procedural programming.	C++ is known as hybrid language because C++ supports both procedural and object oriented programming paradigms.
Data and functions are separated in C because it is a procedural programming language.	Data and functions are encapsulated together in form of an object in C++.

C	C++
C does not support information hiding.	Data is hidden by the Encapsulation to ensure that data structures and operators are used as intended.
Built-in data types is supported in C.	Built-in & user-defined data types is supported in C++.
C is a function driven language because C is a procedural programming language.	C++ is an object driven language because it is an object oriented programming.
Function and operator overloading is not supported in C.	Function and operator overloading is supported by C++.
C is a function-driven language.	C++ is an object-driven language
Namespace features are not present inside the C.	Namespace is used by C++, which avoid name collisions.
Header file used by C is stdio.h.	Header file used by C++ is iostream.h.
Reference variables are not supported by C.	Reference variables are supported by C++.
Virtual and friend functions are not supported by C.	Virtual and friend functions are supported by C++.
C does not support inheritance.	C++ supports inheritance.
Instead of focusing on data, C focuses on method or process.	C++ focuses on data instead of focusing on method or procedure.
C provides malloc() and calloc() functions for dynamic memory allocation, and free() for memory de-allocation.	C++ provides new operator for memory allocation and delete operator for memory de-allocation.
Direct support for exception handling is not supported by C.	Exception handling is supported by C++.
scanf() and printf() functions are used for input/output in C.	cin and cout are used for input/output in C++.

Basic Concepts of Object Oriented Programming



1. 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. Objects take up space in the memory and have an associated address.

When a program is executed, the objects interact by sending messages to one another. For example, if "customer" and "account" are to object in a program, then the customer object may send a message to the count object requesting for the bank balance. Each object contain data, and code to manipulate data.

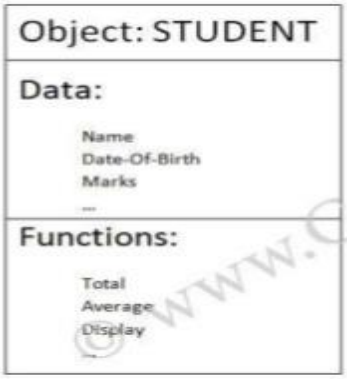


Fig Representation of an object

2. Classes

Class is a user-defined data type which contains the entire set of data and code of an object . 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 similar types. For examples, Mango, Apple and orange members of class fruit. Classes are user-defined that types and behave like the built-in types of a programming language. The syntax used to create an object is not different then the syntax used to create an integer object in C. If fruit has been defines as a class, then the statement

Fruit Mango;

Will create an object **mango** belonging to the class **fruit.**

3. 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*.

4. Data Abstraction

Abstraction refers to the act of representing essential features without including the background details or explanation. 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. The attributes are some time called *data members* because they hold information. The functions that operate on these data are sometimes called *methods or member function*

5. Inheritance

Inheritance is the process by which objects of one class acquired the properties of objects of another classes. It supports the concept of *hierarchical classification.* 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 1.6.

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 feature of both the classes. The real appeal and power of the inheritance mechanism is that it Allows the programmer to reuse a class i.e almost, but not exactly, what he wants, and to tailor the class in such a way that it does not introduced any undesirable side-effects into the rest of classes.

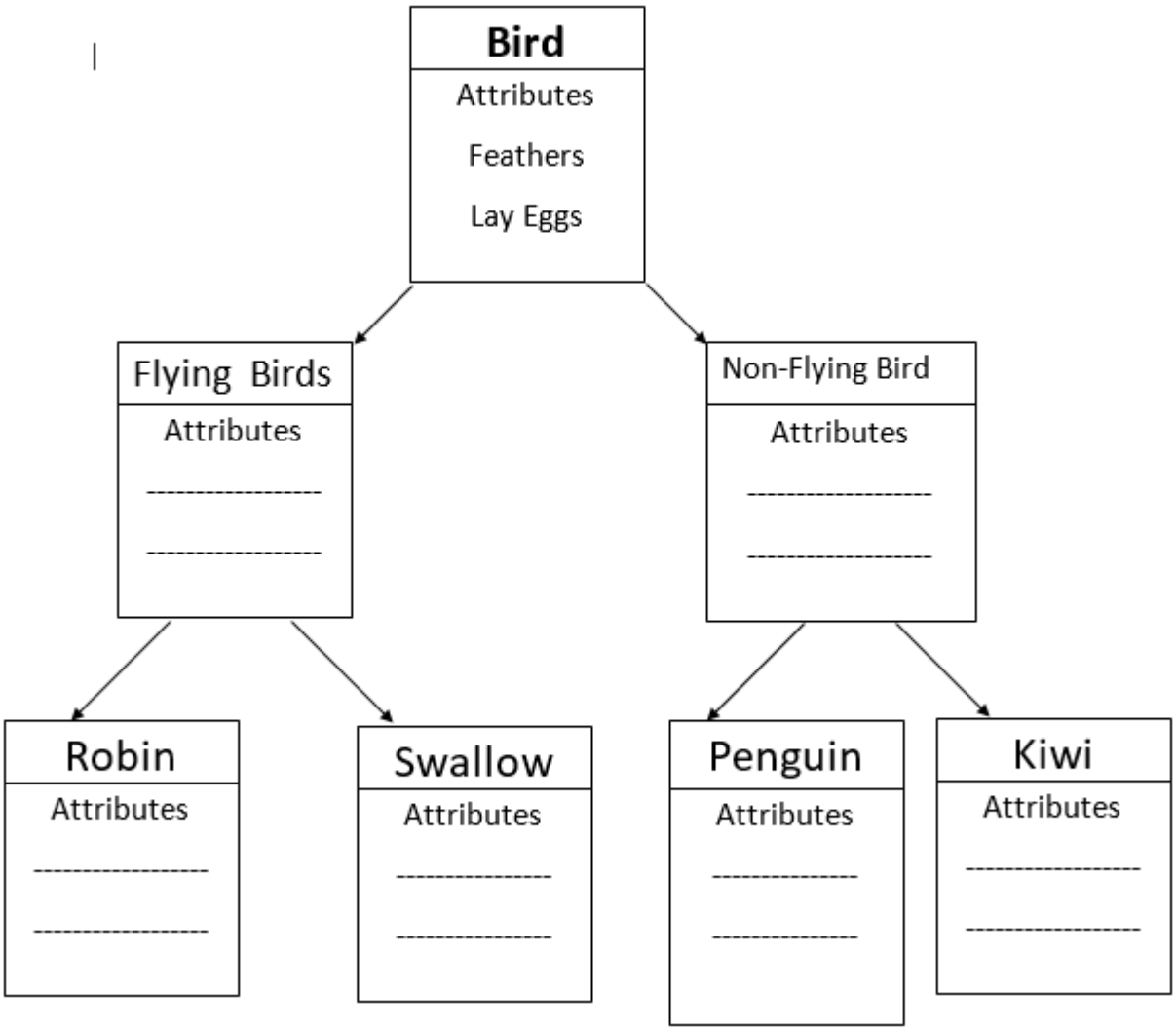


Fig. 1.6 Property inheritances

6. Polymorphism

Polymorphism 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. 1.7 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.*

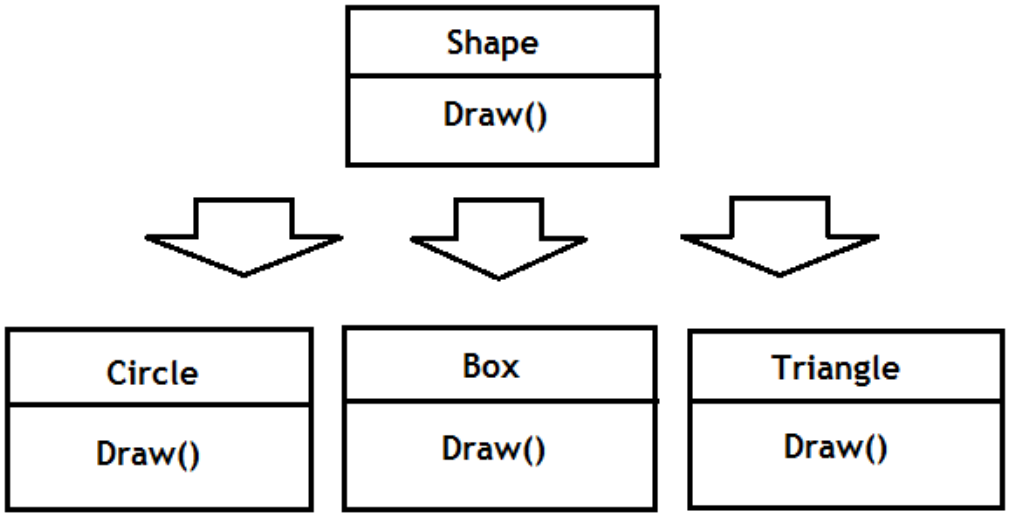


Fig. 1.7 function overloading

7. 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

8. 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.

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

Employee. Salary (name);