Language Paradigms:

- Imperatives – Procedural Programming: C, Pascal, COBOL, FORTRAN etc.
- Applicative – Functional Programming – LISP, ML.
- Rule-based – Logic Programming :– PROLOG
- Object-Oriented Programming: – C++, JAVA, SMALLTALK

Our focus here is on procedural and Object oriented Programming approach.

Procedural programming Language

In procedural programming programs are organized in the form of subroutines. The subroutines do not let code duplication. This technique is only suitable for medium sized software applications. Conventional programming, using HLL e.g. COBOL, FORTRAN, C etc is commonly known as procedural programming. A program in Procedural Language is a list of instructions each statement in the language tells the computer to do something involving – reading, calculating, writing output. A number of functions are written to accomplish such tasks. Program become larger, it is broken into smaller units – functions. The primary focus of procedural oriented programming is on functions rather than data.

Procedure oriented programming basically consists of writing a list of instructions for computer to follow, and organize these instructions into groups known as functions.

In procedural approach,

- A program is a list of instruction.
- When program in PL become larger, they are divided into functions(subroutines, sub-programs, procedures).
- Functions are grouped into modules.

- **Problems with Structured Programming:** As programs grow larger, even structured programming approach begins to show signs of strain. No matter how well the structured programming approach is implemented, the project becomes too complex, the schedule slips, more programmers are needed, and costs skyrocket.
- **Data Undervalued:** Data is given second-class status in the organization of procedural languages. A global data can be corrupted by functions that have no business changing it. In addition, since many functions access the same global data, the way the data is stored becomes critical.
- **Relationship to the Real World:** Procedural programs are often difficult to design because their chief components – functions and data structures – don't model the real world very well.
- **New Data Types:** It is difficult to create new data types with procedural languages. Furthermore, most Procedural languages are not usually extensible and hence procedural programs are more complex to write and maintain.

Structured Programming

Structured programming (sometimes known as *modular programming*) is a subset of procedural programming that enforces a top-down design model, in which developers map out the overall program structure into separate subsections to make programs more efficient and easier to understand and modify.

A defined function or set of similar functions is coded in a separate module or submodule, which means that code can be loaded into memory more efficiently and that modules can be reused in other programs.

- Structured programming uses *control structures* to control the flow of statement execution in a program.
- The control structures of a programming language let us combine individual statements into a single program entity with one entry point and one exit point.
- We can then write a program as a sequence of control structures rather than a sequence of individual statements.

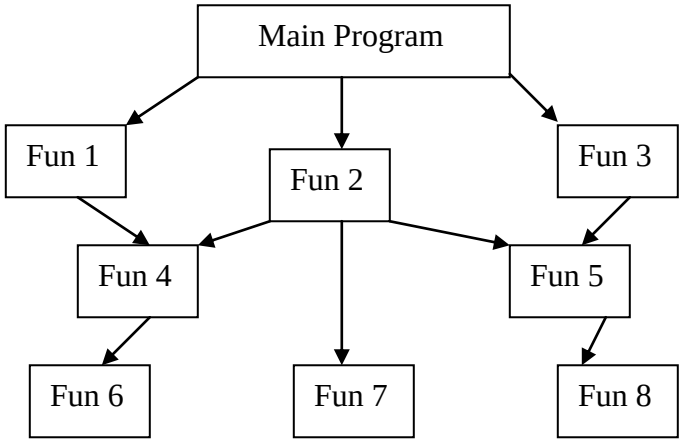
There are three categories of control structures:
1. **Sequence**
2. **Selection** (Decision)
3. **Iteration** (Repetition)

In C++, sequence is illustrated by *compound statements*.

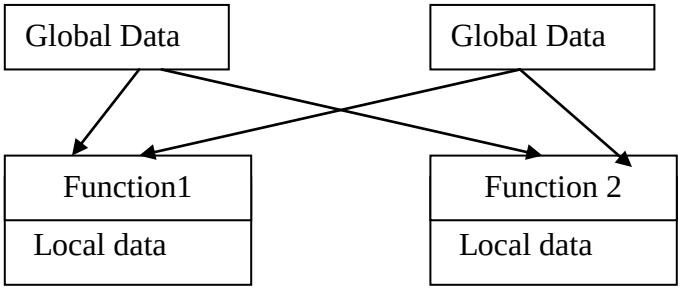
- A compound statement is a group of statements enclosed by braces. Control flows from statement 1 to statement 2 and so on.
- A structure program proceeds from the original problem at the top level down to its detailed sub-problems at the bottom level. This is called a *top-down* approach.

..

A pictorial views



– In a multi function program, two types of date are used: local and global.
  – Data items are placed as global so that they can be accessed by all the functions freely.
  – Each function may have their own data also called as local data.



Some points characterizing Procedural approach:

- Emphasis is on doing things, (Algorithms)
- Large programs are divided into smaller program – function
- Most of functions share global data.
- Data more openly around system from function to function.
- Function transform data from one form to another
- Employs top-down approach for program design

Limitation of Procedural language

- In large program, it is difficult to identify which data is used for which function.
- Global variable overcome the local variable.
- To revise an external data structure, all functions that access the data should also be revised.
- Maintaining and enhancing program code is still difficult because of global data.
- Focus on functions rather than data.
- It does not model real world problem very well. Since functions are action oriented and do not really correspond to the elements of problem.

The Object Oriented Programming Approach:

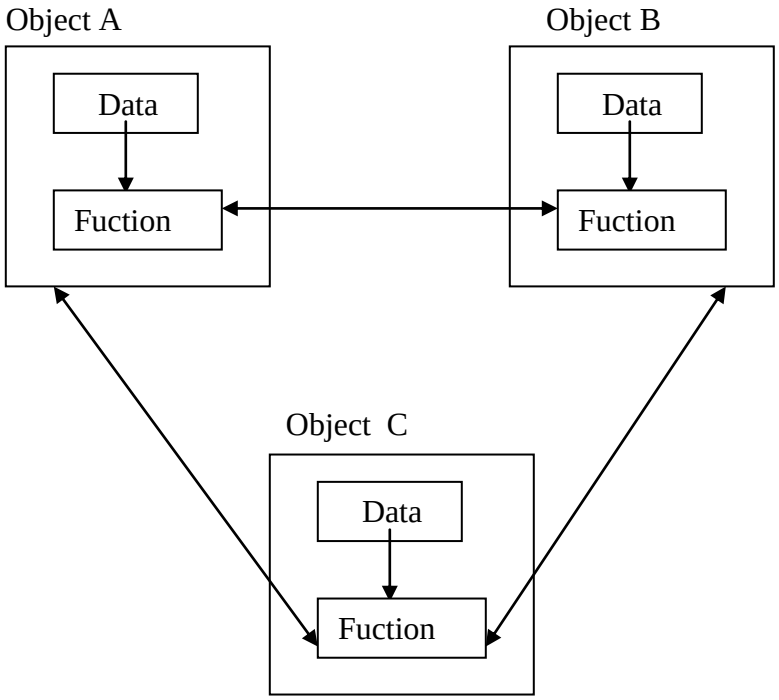
Object-Oriented programming is a programming methodology that associates data structures with a set of operators which act upon it. The fundamental concept behind object-oriented programming is to combine or *encapsulate* both *data* (or *instance variables*) and *functions* (or *methods*) that operate on that data into a single unit. This unit is called an *object*. The data is hidden, so it is safe from accidental alteration. An object's functions typically provide the only way to access its data. In order to access the data in an object, we should know exactly what functions interact with it. No other functions can access the data. Hence OOP focuses on data portion rather than the process of solving the problem. In other words, OOP is a method of implementation in which programs are organized as co-operative collections of objects.

An object-oriented program typically consists of a number of objects, which communicate with each other by calling one another's functions. This is called *sending a message* to the object. This kind of relation is provided with the help of communication between two objects and this communication is done through information called *message*.

In addition, object-oriented programming supports *encapsulation*, *abstraction*, *inheritance*, and *polymorphism* to write programs efficiently. Examples of object-oriented languages include *Simula*, *Smalltalk*, *C++*, *Python*, *C#*, *Visual Basic .NET* and *Java* etc.

Generally in Object Oriented programming

- Emphasis is on data rather than procedures.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects .
- Functions & data are tied together in the data structures so that data abstraction is introduced in addition to procedural abstraction.
- Data is hidden & can't be accessed by external functions.
- Object can communicate with each other through function.
- New data & functions can be easily added.
- Follows Bottom up approach.



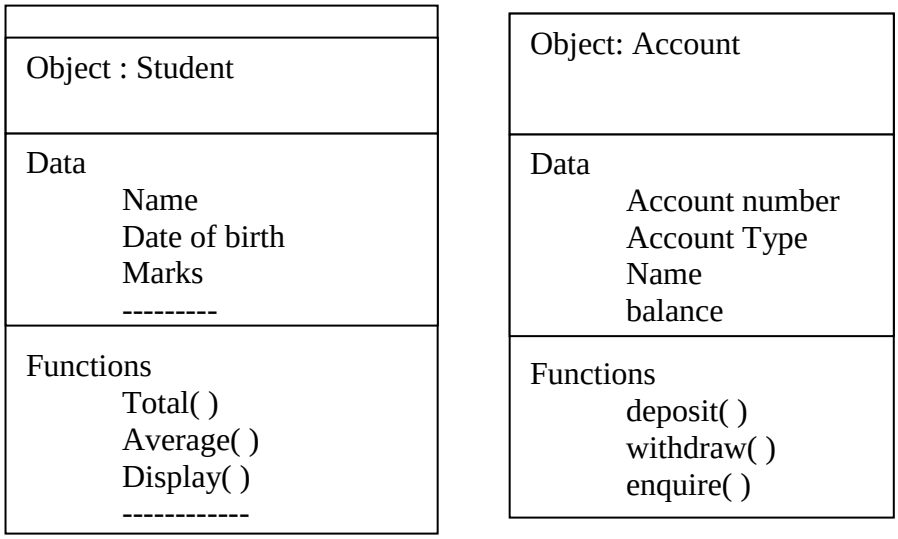
Features of Object Oriented Language:

1. **Objects and Class**: The concept of *classes* and *objects* is the heart of the OOP. Informally, objects are the entities in an object oriented system through which we perceive the world around us. We naturally see our environment as being composed of things which have recognizable identities & behavior. The entities are then represented as objects in the program. They may represent a person, a place, a bank account, or any item that the program must handle. For example Automobiles are objects as they have size, weight, color etc as attributes (ie data) and starting, pressing the brake, turning the wheel, pressing accelerator pedal etc as operation (that is functions).

   A class is a framework that specifies what data and what functions will be included in objects of that class. It serves as a plan or blueprint from which individual objects are created. A Class is the well description of the object. It defines a data type, much like a **struct** in C programming language and built in data type(int char, float etc).

   Defining class doesn't create an object but class is the description of object's attributes and behaviors. Object is an instance of a class just like variable of structure in C.

   Person Class : Attributes: Name, Age, Sex etc.
   Behaviors: Speak(), Listen(), Walk()
   Vehicle Class: Attributes: Name, model, color, height etc



Example of objects:
Physical Objects:
– Automobiles in traffic flow. simulation
– Countries in Economic model
– Air craft in traffic – control system .
Computer user environment objects.
-Window, menus, icons etc
Data storage constructs.
-Stacks, Trees etc
Human entities:
-employees, student, teacher etc.
Geometric objects:
-point, line, Triangle etc.
Objects mainly serve the following purposes:
- Understanding the real world and a practical base for designers
- Decomposition of a problem into objects depends on the nature of problem.

When class is defined, objects are created as

<classname> <objectname> ;

If employee has been defined as a class, then the statement
employee manager;
Will create an object manager belonging to the class employee.

Each class describes a possibly infinite set of individual objects, each object is said to be an instance of its class and each instance of the class has its own value for each attribute but shares the attribute name and operations with other instances of the class. The following points gives the idea of class:
- A class is a template that unites data and operations.
- A class is an abstraction of the real world entities with similar properties.
- Ideally, the class is an implementation of abstract data type.

2. **Abstraction:** Abstraction is the essence of OOP. Abstraction means the representation of the essential features without providing the internal details and complexities. In OOP, abstraction is

achieved by the help of class, where data and methods are combined to extract the essential features only. For example, if we represent chair as an object it is sufficient for us to understand that the purpose of chair is to sit. So we can hide the details of construction of chair and things required to build the chair. It focuses the outside view of an object, separating its essential behavior from its implementation.

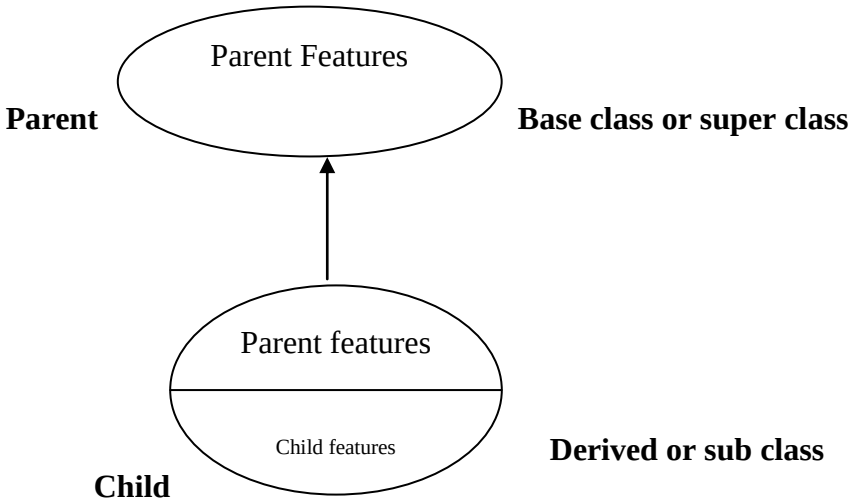
The class is a construct in C++ for creating user-defined data types called Abstract Data Types (ADT)

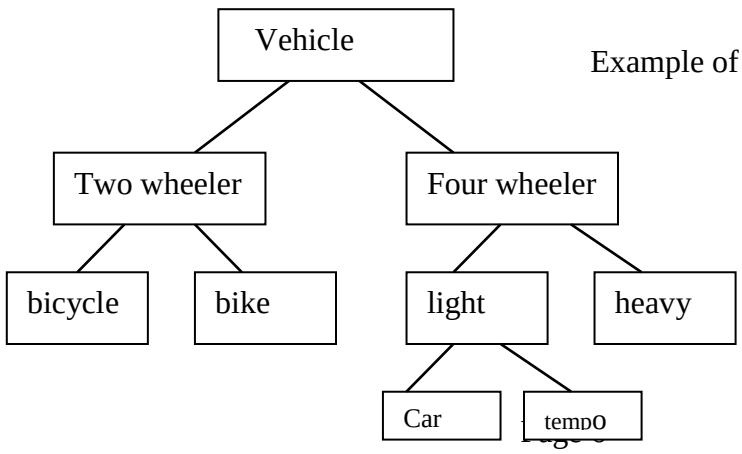
3. **Encapsulation:** Encapsulation is the process of combining the data (called fields or attributes) and functions (called methods or behaviors) into a single framework called class. Encapsulation helps preventing the modification of data from outside the class by properly assigning the access privilege to the data inside the class. So the term ***data hiding*** is possible due to the concept of encapsulation, since the data are hidden from the outside world. so that it is safe from accidental alteration.

4. Inheritance:

Inheritance is the process by which objects of one class acquire the characteristics of object of another class. In OOP, the concept of inheritance provides the idea of **reusability**. We can use additional features to an existing class without modifying it. This is possible by deriving a new class (derived class) from the existing one (base class).This process of deriving a new class from the existing base class is called inheritance .

It supports the concept of hierarchical classification. It allows the extension and reuse of existing code without having to rewrite the code.



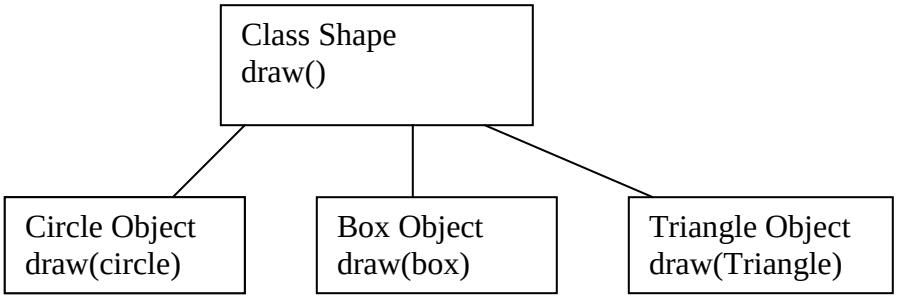


Example of Inheritance class hierarchy

5. **Polymorphism:** Polymorphism means the quality of having more than one form. The representation of different behaviors using the same name is called polymorphism. However the behavior depends upon the attribute the name holds at particular moment. For example if we have the behavior called communicate for the objects vertebrates, then the communicate behavior applied to objects say dogs is quite different from the communicate behavior to objects human. So here the same name communicate is used in multiple process one for human communication, what we simply call talking. The other is used fro communication among dogs, we may say barking, definitely not talking.

Polymorphism is important when object oriented programs dynamically creating and destroying the objects in runtime. Example of polymorphism in OOP is operator overloading, function overloading.

For example operator symbol '+' is used for arithmetic operation between two numbers, however by overloading (means given additional job) it can be used over Complex Object like currency that has Rs and Paisa as its attributes, complex number that has real part and imaginary part as attributes. By overloading same operator '+' can be used for different purpose like concatenation of strings.



Why do We Need Object-Oriented Programming?

OOP was developed because limitations were discovered in earlier approaches to programming.

In addition, object-oriented programming supports *encapsulation*, *abstraction*, *inheritance*, and *polymorphism* to write programs efficiently.

Advantages of OOPs

Object oriented programming contributes greater programmer productivity, better quality of software and lesser maintenance cost. The main advantages are:

- Making the use of inheritance, redundant code is eliminated and the existing class is extended.
- Through data hiding, programmer can build secure programs that cannot be invaded by code in other pats of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- System can be easily upgraded from small to large systems.
- Software complexity can be easily managed.
- Message passing technique for communication between objects makes the interface description with external system much simpler.
- Aids trapping in an existing pattern of human thought into programming.
- Code reusability is much easier than conventional programming languages.

Disadvantages of OOPs

- Compiler and runtime overhead. Object oriented program required greater processing overhead – demands more resources.
- An object's natural environment is in RAM as a dynamic entity but traditional data storage in files or databases
- Re-orientation of software developer to object-oriented thinking.
- Requires the mastery in software engineering and programming methodology.
- Benefits only in long run while managing large software projects.
- The message passing between many objects in a complex application can be difficult to trace & debug.

Object oriented Languages:

The language should support several of OOPs concepts to claim that they are Object-Oriented. Depending upon the features they support, they can be classified as

1. Object–Based Languages
2. Object-Oriented Languages

Object Based Language: Supports encapsulation & object identity without supporting the important features of OOP languages such as Inheritance and Dynamic Bindings. Major features are

- Data Encapsulation
- Data hiding & access mechanisms
- Automatic initialization & clear-up of objects
- Operator overloading
- Don't support inheritance & Dynamic binding . e.g Ada, Visual Basic

Object-Based Language : Encapsulation + Object Identity

Object – Oriented language : Incorporates all features of object-based language plus inheritance and dynamic bindings

Object-Oriented L : Object based feature + inheritance + dynamic bindings

e.g SMALLTALK , C++ , JAVA,EIFFEL etc.

Application of OOP

Applications of OOP are beginning to gain importance in many areas.

- The most popular application of OOP, up to now. Has been area of user interface design such as Windows .
- Real Business systems are often much more complex attributes & behaviors .
- OOP can simplify such complex problem. The areas of application of OOP include
- Real time systems
- Simulation & modeling
- Object-Oriented databases
- Hypertext, hypermedia
- AI & expert system
- Neural Networks & parallel programming
- Decision support & Office automation system
- CAM/CAD systems .
- Computer Based training and Educational Systems