OBJECT ORIENTED PROGRAMMING

Programming Paradigm: A programming paradigm defines the methodology of designing and implementing programs using the key features and building blocks of a programming language. Following are the different programming paradigms

(i) Procedural Programming
(ii) Object Based Programming
(iii) Object Oriented Programming

Procedural Programming: Procedural programming paradigm lays more emphasis on procedure or the algorithm. Data is considered secondary. Data is loosely available to many functions. This paradigm decides which procedure we want; use the best algorithm we can find.

Disadvantages of Procedural programming:

(i) Procedural Programming is susceptible to design changes.
(ii) It leads to increased time and cost overheads during design changes.
(iii) Unable to represent real world relationship that exists among objects.
(iv) The arrangement of the data can't be changed without modifying all the functions that access it.

Object Based Programming: In object-based programming, data and its associated meaningful functions are enclosed in one single entity a *class*. Classes enforce information hiding and abstraction thereby separating the implementation details and the user interface. It also supports user-defined types.

Object Oriented Programming: On OOP, the program is organized around the data being operated upon rather than operations performed. The basic idea behind the OOP is to combine both, data and its functions that operate on the data into a single unit called object. It simplifies the software development and maintenance by providing some concepts/characteristics of OOP:

(i) Data Abstraction
(ii) Data Encapsulation
(iii) Modularity
(iv) Inheritance
(v) Polymorphism

Data Abstraction: Data abstraction means, providing only essential information to the outside world and hiding background details i.e. to represent the needed information in program without presenting the details.

- Abstraction is implemented through public members of a class, as the outside world is given only the essential and necessary information through public members, rest of the things remain hidden.

Data Encapsulation: The wrapping up of data and operations/functions into a single unit (called class) is known as Encapsulation.

- Encapsulation is a way of implementing abstraction.
- Encapsulation is implemented with the help of a class as class binds together data and its associated function under one unit.

Modularity: Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules. The act of partitioning a program into individual components is called modularity.

Inheritance: Inheritance is the capability of one class of things to inherit capabilities or properties from another class. Inheritance enables us to create new classes that reuse, extend, and modify the behaviour that is defined in other classes.

- The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class.
- Inheritance is transitive in nature. It states that if a class A inherits properties from its base class B then all its subclasses will also be inheriting properties of base class A i.e. B.
- When we define a class to derive from another class, the derived class implicitly gains all the members of the base class, except for its constructors and destructors.

Advantages of inheritance:

- Inheritance ensures the closeness with the real-world models.
- Allows the code to be reused as many times as needed. The base class once defined and once it is compiled, it need not be reworked.
- We can extend the already made classes by adding some new features.
- Inheritance is capable of simulating the transitive nature of real-world's inheritance, which in turn saves on modification time and efforts, if required.

E.g.
```
class Student{
        Protected:
                char name[25];
                int rollno;
        public:
                void readStudent();
                void showStudent();
};
class GraduateStudent: protected Student{
        protected:
                char subjects[50];
        public:
                void readGradStud();
                void showGradStud();
};
```

Polymorphism: Polymorphism is the ability for a message or data to be processed in more than one form. In C++, we use Function overloading and Function overriding to achieve polymorphism.

- A function name having several definitions that are differentiable by the number of types of their arguments, is known as function overloading.

Advantages of OOPs:

- OOPs is closer to real world model.
- Hierarchical relationship among objects can be well-represented through inheritance.
- Data can be made hidden or public as per the need. Only the necessary data is exposed enhancing the data security.
- Increased modularity adds ease to program development.
- Private data is accessible only through designed interface in a way suited to the program.

Eg.
```
float area(float a){
        return a*a;
}
float area(float a, float b){
        return a*b;
}
```