Unit 1 : Fundamental of OOP

Syllabus :
Object-Oriented Programming Concepts: Introduction, comparison between procedural programming paradigm and object-oriented programming paradigm, basic concepts of object oriented programming — concepts of an object and a class, interface and implementation of a class, operations on objects, relationship among objects, abstraction, encapsulation, data hiding, inheritance, overloading, polymorphism, messaging.
Classes and Objects: Specifying a class, creating class objects, accessing class members, access specifiers, static members, use of const keyword, friends of a class, empty classes, nested classes, local classes, abstract classes, container classes, bit fields and classes.

Section 1: Object-Oriented Programming Concepts

1.1 Introduction to OOP

Object-Oriented Programming (OOP) is a programming paradigm that uses classes and objects for designing and implementing software. It is based on the principles of encapsulation, inheritance, abstraction and polymorphism. It emphasizes the encapsulation of data (variable) and behavior (function) within objects, promoting reusability, modularity, and maintainability.

A. Benefits of OOP

1. **Improved code organization:** OOP organizes code around objects and classes, making it easier to understand and maintain.
2. **Modularity:** OOP promotes the development of modular components, allowing for easier troubleshooting and updates.
3. **Reusability:** Objects can be reused across different parts of a program or in different programs, leading to faster development and fewer errors.

B. Procedural vs. Object-Oriented Programming

Aspect	Procedural Programming	Object-Oriented Programming
Primary Focus	Focuses on functions and procedure.	Focuses on classes and objects that encapsulate data and behavior.
Data Handling	Data is typically global and can be accessed by any part of the program.	Emphasizes encapsulation, data hiding, and object-oriented design principles. Data hiding is achieved through access modifiers.
Organization	Programs are structured around functions.	Programs are structured around objects and classes, promoting modularity and code reusability.

Code Reusability	Code reuse is limited to functions.	Encourages the reuse of classes and objects through inheritance and composition.
Maintainabillity	Managing complexity becomes challenging as the program grows in size.	Provides better organization and easier maintenance through modularization and encapsulation.
Encapsulation	Not a primary concern; data and functions are loosely connected.	Encourages bundling data and methods into a single unit (class), promoting data hiding and abstraction.
Inheritance	Not supported or limited (e.g., through library functions).	Supports inheritance, allowing new classes to inherit properties and behaviors from existing classes, promoting code reusability.
Polymorphism	Achieved through functions with the same name but different parameters.	Achieved through method overriding and dynamic dispatch, allowing objects to be treated as instances of their parent class.
Abstraction	Limited; focuses more on procedural logic.	Promotes abstraction by simplifying complex systems through modeling classes based on essential features.
Example Languages	C, Pascal, BASIC	Java, C++, Python

1.2 Basic Concept of OOP

A. Classes and Objects:

Classes:

- A class is a blueprint or template for creating objects. It defines the properties (attributes) and behaviors (methods) that objects of that class will have.
- A class encapsulates data members (attributes) and member functions (methods) that operate on the data.
- Classes facilitate code reusability and promote modularity by organizing related data and functions into a single unit.
- Classes act as user-defined data types.
- In C++, a class is declared using the class keyword, followed by the class name and a pair of curly braces containing class members.
- **Syntax:**

```
class MyClass {
    // Class members (attributes and methods)
};
```