MODULE : 4.1(C++ Basic)

**Q.1> What is OOP? List OOP concept**

1. OOP stands for Object-Oriented Programming. It is a programming paradigm that uses "objects" to represent real-world entities and their interactions. The main concepts of OOP include:
2. **Class**: A blueprint or template that defines the structure and behavior of objects. It encapsulates data and methods that operate on that data.
3. **Object**: An instance of a class, representing a specific realization of the class blueprint.
4. **Encapsulation**: The concept of bundling data (attributes) and methods (functions) that operate on the data within a single unit (the class). It hides the internal details of the object from the outside, and only exposes a public interface to interact with it.
5. **Abstraction**: Abstraction allows you to represent the essential features of an object while hiding the unnecessary details. It focuses on what an object does rather than how it does it.
6. **Inheritance**: The process by which one class acquires the properties and behaviors of another class. It promotes code reuse and helps create a hierarchical relationship between classes.
7. **Polymorphism**: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to represent different types of objects.

In C++, specific OOP concepts are implemented as follows:

* **Class**: Defines the blueprint for creating objects, including their attributes and member functions.
* **Object**: An instance of a class created using the "new" keyword or as a stack variable.
* **Encapsulation**: Achieved by using access specifiers like "public," "private," and "protected" to control the visibility of class members.
* **Abstraction**: Achieved by defining pure virtual functions (abstract methods) and creating abstract classes using the "virtual" keyword.
* **Inheritance**: Declared using the ":" symbol after the class name, followed by the access specifier and the base class from which you want to inherit.
* **Polymorphism**: Achieved through virtual functions and function overriding. When a base class function is declared virtual, it can be overridden in derived classes.

**Q.1 What is the difference between OOP and POP?**

* OOP (Object-Oriented Programming) and POP (Procedural-Oriented Programming) are two different programming paradigms, each with its own approach to organizing and structuring code. C++ is a versatile language that supports both paradigms, but it is often associated more closely with OOP due to its extensive support for object-oriented features. Let's explore the key differences between OOP and POP:

1. **Programming Paradigm:**
   * OOP: In object-oriented programming, the primary focus is on creating and manipulating objects. An object is a self-contained unit that combines data (attributes) and behaviors (methods/functions) that operate on that data. The program's structure is centered around classes and objects, encouraging encapsulation, abstraction, inheritance, and polymorphism.
   * POP: Procedural-oriented programming, on the other hand, is focused on writing procedures or functions that perform operations on data. The emphasis is on the sequence of function calls to achieve the desired outcome. In POP, data and functions are treated separately, and there is less emphasis on encapsulation and abstraction.
2. **Data and Function Organization:**
   * OOP: In OOP, data and functions that operate on that data are encapsulated within classes. Objects are instances of these classes, and they interact with each other through methods defined in those classes.
   * POP: In POP, data and functions are separate. Functions are written to perform specific tasks on data, which might be organized in global variables or passed between functions as parameters.
3. **Inheritance:**
   * OOP: Object-oriented programming allows for the concept of inheritance, which enables one class (a subclass or derived class) to inherit the properties and behaviors of another class (a superclass or base class). This promotes code reusability and helps create a hierarchy of classes.
   * POP: Procedural programming does not inherently support inheritance. Code reuse is generally achieved by using functions, and there is no formal mechanism for class hierarchy.
4. **Polymorphism:**
   * OOP: Polymorphism allows objects of different classes to be treated as objects of a common base class during runtime. It enables dynamic dispatch of methods and is essential for achieving flexibility and extensibility in code.
   * POP: Polymorphism is not directly supported in procedural programming, as it is a feature more closely associated with OOP.
5. **Program Structure:**
   * OOP: The structure of OOP programs revolves around objects and their interactions, leading to modular and organized code.
   * POP: Procedural programming focuses on functions and the flow of the program, which may result in longer and more complex code when handling large projects.