**MODULE: 4**

**OOPS Concept**

**Q1. WAP to print “Hello World” using C++**

**A1.** #include<iostream>

using namespace std;

int main()

{

cout<<"Hello World!"<<endl;

}

**Q2. What is OOP? List OOP concepts**

**A2.** Object-Oriented Programming (OOP) is a programming paradigm that uses "objects" to design applications and computer programs. Objects are instances of classes, which are blueprints for creating objects. This approach helps in organizing complex programs, making them easier to manage, maintain, and extend.

**KEY CONCEPTS OF OOP:**

1. **Class & Object**

A Class is a blueprint of an Object. It defines a set of attributes and methods that the created objects can have. An object is an instance of a class. It is a self-contained entity that consists of data and methods to manipulate the data.

1. **Inheritance**

Inheritance is the mechanism by which one class can inherit the attributes and methods of another class. It provides a way to create a new class from an existing class.

1. **Polymorphism**

Polymorphism allows methods to do different things based on the object it is acting upon, even though they share the same name.

1. **Encapsulation**

Encapsulation is the concept of wrapping data (attributes) and methods (functions) into a single unit, i.e., a class.

1. **Abstraction**

Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object.

1. **Constructor & Destructor**

A constructor is a special type of method that is automatically called when an object is created. It is used to initialize the object's attributes. A destructor, on the other hand, is called when an object is destroyed. It is used to clean up resources that the object may have acquired during its lifetime.

**Q3. What is the difference between OOP and POP?**

**Q3.**

| **OOP** | **POP** |
| --- | --- |
| Object Oriented Programming. | Procedure Oriented Programming. |
| It is Object Oriented. | It is Structured Oriented. |
| The program is divided into Objects. | The program is divided into Functions. |
| Follows Bottom-Up approach. | Follows the Top-Down approach. |
| Inheritance property is used. | Inheritance is not allowed. |
| It uses access specifiers. | It doesn’t use access specifiers. |
| Adding new data & functions is easy. | Expanding new data & functions is not easy. |
| The existing code is reusable. | No code reuseability. |
| Used for solving big problems. | Not suitable for solving big problems. |
| Eg: C++, Java | Eg: C, Pascal |