**Module 3 Introduction to OOPS Programming**

**Q-1.Introduction to C++**

**THEORY EXERCISE:**

1. **What are the key differences between Procedural Programming and Object-Oriented Programming (OOP) ?**

* **Procedural Programming (POP) focuses on functions and procedures, following a top-down approach. In contrast, OOP organizes code into objects and classes, using a bottom-up approach. POP lacks features like inheritance and encapsulation, while OOP supports them, making code more modular and reusable. OOP also promotes data hiding and abstraction.**

**2.List and explain the main advantages of OOP over POP ?**

* **OOP offers better code reusability through inheritance and promotes data security via encapsulation. It simplifies complex programs by modeling real-world entities using objects. OOP supports modularity, making programs easier to debug, maintain, and scale. It also enhances code flexibility through polymorphism.**

**3. Explain the steps involved in setting up a C++ development environment.**

* **First, install a C++ compiler like GCC or MSVC. Then, install an IDE or text editor such as Code::Blocks, Dev C++, or VS Code. Configure the compiler in the IDE settings. Optionally, set environment variables if using command line tools. Finally, write and run your first program to test the setup.**

**4. What are the main input/output operations in C++? Provide examples?**

* **C++ uses cin for input and cout for output, both from the <iostream> header. For example: cpp , Copy Edit**

**int age;**

**cin>> age; // Input**

**cout << "Age is: " << age; // Output**

**Q-2. Variables, Data Types, and Operators**

**THEORY EXERCISE:**

1. **What are the different data types available in C++? Explain with examples?**
   * In C++, data types define the type of data a variable can store, helping the compiler allocate memory and determine the operations allowed on that data. They are mainly classified into fundamental, derived, and user-defined types**.**

**(A).Fundamental (Primitive) Data Types – Basic built-in types:**

* + int – Stores whole numbers (e.g., int age = 25;)
  + float – Stores decimal numbers with single precision (e.g., float price = 99.5;)
  + double – Stores decimal numbers with double precision (e.g., double pi = 3.14159;)
  + char – Stores a single character (e.g., char grade = 'A';)
  + bool – Stores true or false (e.g., bool isPassed = true;)
  + void – Represents no value (used for functions without a return type)

**(B).Derived Data Types – Formed from fundamental types:**

* Array – Stores multiple elements of the same type (e.g., int marks[5] = {90, 85, 88, 92, 95};)
* Pointer – Stores memory address of a variable (e.g., int\* ptr = &age;)
* Reference – Alias for another variable (e.g., int& ref = age;)

**(C).User-defined Data Types – Created by the programmer:**

* struct – Groups related variables (e.g., struct Student { int roll; char name[20]; };)
* class – Defines objects with attributes and methods (OOP concept)
* enum – Defines a set of named constants (e.g., enum Day { Mon, Tue, Wed };)

**Example:-**

**#include <iostream>**

**using namespace std;**

**main() {**

**int age = 20; // int type**

**float height = 5.9; // float type**

**char grade = 'A'; // char type**

**bool isPassed = true; // bool type**

**double pi = 3.14159265; // double type**

**cout << "Age: " << age << "\nHeight: " << height**

**<< "\nGrade: " << grade << "\nPassed: " << isPassed**

**<< "\nPi: " << pi;**

**}**

**2. Explain the difference between implicit and explicit type conversion in C++?**

**3. What are the different types of operators in C++? Provide examples of each?**

**4. Explain the purpose and use of constants and literals in C++?**