**SE ASSIGNMENT 2**

**OOPS Concept**

Lab Exercise:

1. First C++ Program: Hello World o Write a simple C++ program to display "Hello, World!". o Objective: Understand the basic structure of a C++ program, including #include, main(), and cout.

Ans. #include <iostream>

using namespace std;

int main() {

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

return 0;

}

1. Basic Input/Output

o Write a C++ program that accepts user input for their name and age and then displays a personalized greeting.

o Objective: Practice input/output operations using cin and cout.

Ans. #include <iostream>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

getline(cin, name);

cout << "Enter your age: ";

cin >> age;

cout << "Hello, " << name << "! You are " << age << " years old." <<endl;

return 0;

}

3. POP vs. OOP Comparison Program

o Write two small programs: one using Procedural Programming (POP) to calculate the area of a rectangle, and another using Object-Oriented Programming (OOP) with a class and object for the same task.

o Objective: Highlight the difference between POP and OOP approaches.

Ans. #include <iostream>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

getline(cin, name);

cout << "Enter your age: ";

cin >> age;

cout << "Hello, " << name << "! You are " << age << " years old." <<endl;

return 0;

}

4. Setting Up Development Environment

o Write a program that asks for two numbers and displays their sum. Ensure this is done after setting up the IDE (like Dev C++ or CodeBlocks).

o Objective: Help students understand how to install, configure, and run programs inan IDE.

Ans. #include <iostream>

using namespace std;

class Rectangle {

private:

float width;

float height;

public:

Rectangle(float w, float h) : width(w), height(h) {}

float calculateArea() {

return width \* height;

}

void displayDetails() {

cout << "Width: " << width << endl;

cout << "Height: " << height << endl;

cout << "Area: " << calculateArea() << endl;

}

};

int main() {

float width, height;

cout << "Enter the width of the rectangle: ";

cin >> width;

cout << "Enter the height of the rectangle: ";

cin >> height;

Rectangle rect(width, height);

rect.displayDetails();

return 0;

}

Theory Exercise:

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

Ans. 1.Focus and Structure:

Procedural Programming:

Focuses on procedures or routines (functions) that operate on data.

The program is structured around a sequence of procedures or steps.

Object-Oriented Programming:

Focuses on creating objects that encapsulate data and behavior.

The program is structured around classes and objects.

2.Data and Behavior:

Procedural Programming:

Data is passed explicitly between functions.

Functions operate on external data, and there is limited data encapsulation.

Object-Oriented Programming:

Data is encapsulated within objects, and objects expose behaviors through methods.

Objects interact with one another via messages (methods).

3. Abstraction:

Procedural Programming:

Lacks abstraction; operations are performed directly on data.

Object-Oriented Programming:

Emphasizes abstraction by hiding complexity through encapsulation and exposing only necessary parts.

4. Inheritance and Reusability:

Procedural Programming:

Typically lacks inheritance, which means code reuse is limited to functions or procedures.

Object-Oriented Programming:

Supports inheritance, allowing classes to inherit properties and behaviors from other classes.

Promotes reusability and modularity.

5. Encapsulation:

Procedural Programming:

Functions operate on data with minimal attention to encapsulation; data may be global or shared across functions.

Object-Oriented Programming:

Emphasizes encapsulation, hiding internal state and requiring objects to interact through defined interfaces.

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

Ans. OOP:

Emphasis is on data rather than procedure (function).

Programs are divided into objects.

Functions that operate on the data of an object are ties together in the data structure.

Data is hidden and cannot be accessed by external function.

POP:

Large programs are divided into smaller programs known as functions.

Most of the functions share global data.

Data move openly around the system from function to function.

Functions change the value of data at any time from any place.

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

Ans. Choose an IDE :

IDEs: Visual Studio, Code::Blocks, CLion (easy setup).

Editors: VS Code, Sublime Text (needs configuration).

Install a Compiler:

Windows: Install MinGW or Visual Studio (MSVC).

macOS: Install Xcode Command Line Tools (xcode-select --install).

Linux: Install GCC (sudo apt install build-essential).

Configure IDE/Editor:

For IDEs: Set up projects and link to the compiler (automatic in most cases).

Install a Debugger:

GDB (with GCC), LLDB (with Clang), or MSVC Debugger (Visual Studio).

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

Ans. Input Operations:

std::cin: Reads input from the standard input (keyboard).

Output Operations:

std::cout: Outputs data to the standard output (console).

Special Output:

std::cerr: Outputs error messages