Module-3 Introduction to OOPS Programming

1. *Introduction to C++*

**LAB EXERCISES:**

1. First C++ Program: “Hello World

* Write a simple C++ program to display "Hello, World!".
* 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”;

}

1. #include <iostream>
   * This is a preprocessor directive that includes the standard input-output stream library.
   * Needed for cout (output) and cin (input).
2. using namespace std;
   * C++ has namespaces to organize code.
   * This line lets you use standard library names like cout without writing std::cout.
3. int main()
   * This is the main function — the entry point of every C++ program.
   * The program starts executing from here.
4. cout << "Hello, World!";
   * cout stands for "character output" — used to display text on the console.
   * << is the insertion operator that sends data to cout.
5. return 0;
   * Returns 0 to the operating system, meaning the program ended successfully.
6. Basic Input/Output

* Write a C++ program that accepts user input for their name and age and them display a personalized greeting.
* Objective: practice input/output operations using cin and cout.

Ans. #include<iostream>

using namespace std;

int main() {

int age;

string name;

cout << "Enter the name: ";

cin >> name;

cout << "Enter the age: ";

cin >> age;

cout << "hello, " << name << "! you are " << age <<

" years old. " << endl;

}

1. POP vs. OOP Comparison Program

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

Ans. **POP(Procedural Oriented Programming )**

#include<iostream>

using namespace std;

int main() {

int area,length, width;

cout << "Enter the length:";

cin >> length;

cout << "Enter the width:";

cin >> width;

area = length \* width;

cout << "Area of Rectangle: " << area << endl;

}

**OOP(Object-Oriented Programming)**

#include<iostream>

using namespace std;

class Rectangle{

public:

double width, length;

void input(){

cout << "Enter the length: ";

cin >> length;

cout << "Enter the width: ";

cin >> width;

}

double area(){

return length \* width;

}

};

int main(){

Rectangle r1;

r1.input();

cout << "Area of rectangle: " << r1.area() << endl;

}

1. Setting Up Development Environment

* 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).
* Objective: Help students understand how to install, configure, and run programs in an IDE.

Ans. 1. Setting Up the Development Environment

1: Dev C++

1. Download & Install
   * Download Dev C++ from: https://sourceforge.net/projects/orwelldevcpp/
   * Run the installer and choose Typical Installation.
2. Open Dev C++
3. Create New File
   * Go to File → New → Source File.
4. Save File
   * Save it with a .cpp extension (e.g., sum.cpp).

Code::Blocks

1. Download & Install
   * Download Code::Blocks with MinGW from: http://www.codeblocks.org/downloads/
   * Install it with the Full Installation option.
2. Open Code::Blocks
3. Create New Project
   * Go to File → New → Project → Console Application → C++.
4. Follow the Wizard
   * Set the project name and location, then finish.

C++ Program – sum of two number

#include<iostream>

using namespace std;

int main(){

int a, b, sum;

cout << "Enter the a: ";

cin >> a;

cout << "Enter the b: ";

cin >> b;

sum = a + b;

cout << "The sum is a and b is " << sum << endl;

}

How to Run

* Dev C++:
  + Click Execute → Compile & Run (or press F11).
* Code::Blocks:
  + Click Build and Run (or press F9).

*2. Variables, Data Types, and Operator*

1. Variables and Constants

o Write a C++ program that demonstrates the use of variables and constants. Create variables of different data types and perform operations on them.

o Objective: Understand the difference between variables and constants.

Ans #include<iostream>

using namespace std;

int main(){

// Constants decalaration

const string Name = "radharaman ju";

const double PI = 3.14;

// variables

int age = 20;

double height = 5.4;

char grade = 'A';

bool isStudent = true;

cout << "Name : " << Name <<endl;

cout << "Value of PI : " << PI <<endl;

cout << "Age: " << age <<endl;

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

cout << "grade: " << grade << endl;

cout << "Is student? " << isStudent << endl;

age = age + 5;

height = height + 0.1;

cout << "\n After Updating variable: " << endl;

cout << "Updated Age: " <<age<<endl;

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

double radius = 2.0;

double area = PI \* radius \* radius;

cout << "\n Area of cricel with radius " << radius << " is: " <<area << endl;

}

This program:

* Shows **variables** (whose values can change)
* Shows **constants** (whose values cannot change)
* Uses **different data types** (int, double, char, bool, string)
* Performs **operations** and demonstrates the difference clearly.

1. Type Conversion

* Write a C++ program both implicit and explicit type conversion and prints the result.
* Objective: practice type casting in C++.

Ans. #include <iostream>

using namespace std;

int main() {

// ---------- Implicit Type Conversion ----------

int intVal = 10;

double doubleVal = intVal; // int automatically converted to double

cout << "Implicit Type Conversion:" << endl;

cout << "Integer value: " << intVal << endl;

cout << "Converted to double: " << doubleVal << endl;

// When int is divided by double, int is implicitly converted to double

double resultImplicit = intVal / 4.0;

cout << "Result of intVal / 4.0 (implicit conversion): " << resultImplicit << endl;

// ---------- Explicit Type Conversion ----------

double num1 = 9.78;

int num2 = (int)num1; // C-style cast

int num3 = int(num1); // Function-style cast

int num4 = static\_cast<int>(num1); // Modern C++ style cast

cout << "\nExplicit Type Conversion:" << endl;

cout << "Double value: " << num1 << endl;

cout << "C-style cast to int: " << num2 << endl;

cout << "Function-style cast to int: " << num3 << endl;

cout << "static\_cast to int: " << num4 << endl;

// Explicit conversion in a calculation

double resultExplicit = (double)5 / 2; // Forces double division

cout << "Result of (double)5 / 2 (explicit conversion): " << resultExplicit << endl;

return 0;

}

1. Operator Demonstration

* Write a C++ program that demonstrates arithmetic, relational, logical, and bitwise operators. Perform operations using each type of operator and display the results.
* Objective: Reinforce understanding of different types of operators in C++.

Ans. #include <iostream>

using namespace std;

int main() {

int a = 10, b = 5;

// ---------- Arithmetic Operators ----------

cout << "Arithmetic Operators:" << endl;

cout << "a + b = " << (a + b) << endl;

cout << "a - b = " << (a - b) << endl;

cout << "a \* b = " << (a \* b) << endl;

cout << "a / b = " << (a / b) << endl;

cout << "a % b = " << (a % b) << endl;

// ---------- Relational Operators ----------

cout << "\nRelational Operators:" << endl;

cout << "a == b : " << (a == b) << endl;

cout << "a != b : " << (a != b) << endl;

cout << "a > b : " << (a > b) << endl;

cout << "a < b : " << (a < b) << endl;

cout << "a >= b : " << (a >= b) << endl;

cout << "a <= b : " << (a <= b) << endl;

// ---------- Logical Operators ----------

bool x = true, y = false;

cout << "\nLogical Operators:" << endl;

cout << "x && y : " << (x && y) << endl; // Logical AND

cout << "x || y : " << (x || y) << endl; // Logical OR

cout << "!x : " << (!x) << endl; // Logical NOT

// ---------- Bitwise Operators ----------

cout << "\nBitwise Operators (on a=10, b=5):" << endl;

cout << "a & b : " << (a & b) << endl; // Bitwise AND

cout << "a | b : " << (a | b) << endl; // Bitwise OR

cout << "a ^ b : " << (a ^ b) << endl; // Bitwise XOR

cout << "~a : " << (~a) << endl; // Bitwise NOT

cout << "a << 1 : " << (a << 1) << endl; // Left shift

cout << "a >> 1 : " << (a >> 1) << endl; // Right shift

return 0;

}

*3.Control Flow Statements*

1. Write a C++ programs that takes a students marks as input and calculates the grade based on if-else conditions.

Objective: practice conditional statements(if-else).

Ans. #include<iostream>

using namespace std;

int main() {

int marks;

cout << "Enter the marks: ";

cin >> marks;

if(marks>=90){

cout << "Grade A";

} else if(marks>=80){

cout << "Grade B";

} else if(marks>=70){

cout << "Grade C";

} else if(marks>=60) {

cout << "Gade D";

} else {

cout << "Student marks is less then 60";

}

}

2. Number Guessing Game

* Write a C++ program that asks the user to guess number between 1 and 100. The program should provide hints if the guess is too high or too low. Use loops to allow the user multiple attempts.
* Objective: Understand while loops and conditional logic.

Ans. #include <iostream>

#include <cstdlib> // for rand() and srand()

#include <ctime> // for time()

using namespace std;

int main() {

int secretNumber, guess;

// Generate a random number between 1 and 100

srand(time(0)); // seed the random number generator

secretNumber = rand() % 100 + 1;

cout << "Guess the number (between 1 and 100): ";

cin >> guess;

// Loop until the user guesses correctly

while (guess != secretNumber) {

if (guess > secretNumber) {

cout << "Too high! Try again: ";

} else {

cout << "Too low! Try again: ";

}

cin >> guess;

}

cout << "🎉 Congratulations! You guessed the number: " << secretNumber << endl;

return 0;

}

3.Multiplication table

* Write a C++ program to display the multiplication table of a given number using a for loop.
* Objective: Practice using loop.

Ans. #include<iostream>

using namespace std;

int main(){

int i,num;

cout << "Enter the number: ";

cin >> num;

cout << "Multiplication Table of " << num <<":"<< endl;

for(i=1; i<=10; i++){

cout << num << "x" << i << " = " << num \* i << endl;

}

}

1. Nested Control Structures

* Write a program that prints a right-angled triangle using stars (\*) with a nested loop. o Objective: Learn nested loop.
* Objective: Learn nested loop.

Ans. #include<iostream>

using namespace std;

int main(){

int i,j;

for(i=1; i<=5; i++){

for(j=1; j<=i; j++){

cout << "\*";

}

cout << "\n";

}

}

*4. Functions and Scope*

1.Simple Calculator Using Functions

* Write a C++ program that defines functions for basic arithmetic operations (add, subtract, multiply, divide). The main function should call these based on user input.
* Objective: o Objective: Practice defining and using functions in C++;

Ans. #include<iostream>

using namespace std;

double add(double a, double b){

return a+b;

}

double sub(double a, double b){

return a-b;

}

double multi(double a, double b){

return a\*b;

}

double div(double a, double b){

if (b != 0)

return a / b;

else{

cout << "error divison by zero is not allowed " << endl;

return 0;

}

}

int main(){

double num1, num2;

char op;

cout << "Simple calculator" << endl;

cout << "Enter frist number: ";

cin >> num1;

cout << "Enter operator (+,-,\*,/): ";

cin >> op;

cout << "Enter second number: ";

cin >> num2;

double result;

switch (op) {

case '+':

result = add(num1,num2);

cout << "Result: " << result << endl;

break;

case '-':

result = sub(num1,num2);

cout << "Result: " << result << endl;

break;

case '\*':

result = multi(num1,num2);

cout << "Result: " << result << endl;

break;

case '/':

result = div(num1, num2);

cout << "Result: " << result << endl;

break;

default:

cout << "Invalid Operator" << endl;

}

}

2. Factorial calculation using recursion

* Write a C++ program that calculates the factorial of a number using recursion.
* Objective: Understand recursion in functions.

Ans. #include<iostream>

using namespace std;

int factorial(int n) {

if(n==0 || n==1){

return 1;

} else {

return n \* factorial(n-1);

}

}

int main(){

int num;

cout << "Enter a number to find its factorial: ";

cin >> num;

if(num < 0){

cout << "Factorial of a negative number is not defined" << endl;

} else {

cout << "Factorial of" << num << " = " << factorial(num) << endl;

}

}

3. Variable Scope

* Write a program that demonstrates the difference between local and global variables in C++. Use functions to show scope.
* Objective: Reinforce the concept of variable scope.

Ans. #include<iostream>

using namespace std;

int globalVar = 50;

void showScope() {

int localVar = 20;

cout << "Indside Function: " << endl;

cout << "Local variable = " << localVar << endl;

cout << "global variable = " << globalVar << endl;

}

int main() {

int localVar = 10;

cout << "Inside Main:" << endl;

cout << "local variable = " << localVar << endl;

cout << "Global variable = " << globalVar << endl;

cout << "\n Calling function " << endl;

showScope();

}

1. *Arrays and Strings* 
   1. Array Sum and Average

o Write a C++ program that accepts an array of integers, calculates the sum and average, and displays the results.

o Objective: Understand basic array manipulation.

Ans. #include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the number of elements in the array: ";

cin >> n;

int arr[100]; // array size limit (can be increased if needed)

int sum = 0;

// Input array elements

cout << "Enter " << n << " integers:" << endl;

for (int i = 0; i < n; i++) {

cin >> arr[i];

sum += arr[i]; // calculate sum

}

// Calculate average

double average = (double)sum / n;

cout << “sum of array element = “ <<sum << endl;

cout << “Avearge of array elements = “ <<average << endl;

}

* 1. Matrix Addition

o Write a C++ program to perform matrix addition on two 2x2 matrices.

o Objective: Practice multi-dimensional arrays.

Ans. #include <iostream>

using namespace std;

int main() {

int A[2][2], B[2][2], C[2][2];

cout << "Enter elements of first 2x2 matrix (A):" << endl;

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

cin >> A[i][j];

}

}

cout << "Enter elements of second 2x2 matrix (B):" << endl;

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

cin >> B[i][j];

}

}

// Matrix addition

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

C[i][j] = A[i][j] + B[i][j];

}

}

// Display result

cout << "Resultant Matrix (A + B):" << endl;

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

cout << C[i][j] << " ";

}

cout << endl;

}

return 0;

}

* 1. String Palindrome Check

o Write a C++ program to check if a given string is a palindrome (reads the same forwards and backwards).

o Objective: Practice string operations.

Ans. #include <iostream>

#include <string>

using namespace std;

int main() {

string str;

cout << "Enter a string: ";

cin >> str;

int start = 0;

int end = str.length() - 1;

bool isPalindrome = true;

while (start < end) {

if (str[start] != str[end]) {

isPalindrome = false;

break;

}

start++;

end--;

}

if (isPalindrome) {

cout << str << " is a Palindrome." << endl;

} else {

cout << str << " is NOT a Palindrome." << endl;

}

return 0;

}

1. Introduction to Object-Oriented Programming
2. Class for a Simple Calculator

o Write a C++ program that defines a class Calculator with functions for addition, subtraction, multiplication, and division. Create objects to use these functions.

o Objective: Introduce basic class structure.

Ans.

#include <iostream>

using namespace std;

// Class definition

class Calculator {

public:

// Member functions

double add(double a, double b) {

return a + b;

}

double subtract(double a, double b) {

return a - b;

}

double multiply(double a, double b) {

return a \* b;

}

double divide(double a, double b) {

if (b != 0)

return a / b;

else {

cout << "Error! Division by zero is not allowed." << endl;

return 0;

}

}

};

int main() {

Calculator calc; // Object of Calculator

double num1, num2;

char op;

cout << "Simple Calculator using Class" << endl;

cout << "Enter first number: ";

cin >> num1;

cout << "Enter operator (+, -, \*, /): ";

cin >> op;

cout << "Enter second number: ";

cin >> num2;

double result;

// Using class functions

switch (op) {

case '+':

result = calc.add(num1, num2);

cout << "Result: " << result << endl;

break;

case '-':

result = calc.subtract(num1, num2);

cout << "Result: " << result << endl;

break;

case '\*':

result = calc.multiply(num1, num2);

cout << "Result: " << result << endl;

break;

case '/':

result = calc.divide(num1, num2);

cout << "Result: " << result << endl;

break;

default:

cout << "Invalid operator!" << endl;

}

return 0;

}

3. Inheritance Example

o Write a program that implements inheritance using a base class Person and derived classes Student and Teacher. Demonstrate reusability through inheritance.

o Objective: Learn the concept of inheritance.

Ans. #include <iostream>

using namespace std;

// Base class

class Person {

public:

string name;

int age;

void getDetails() {

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

}

void displayDetails() {

cout << "Name: " << name << ", Age: " << age << endl;

}

};

// Derived class: Student

class Student : public Person {

public:

string course;

int rollNo;

void getStudentDetails() {

getDetails(); // reuse base class function

cout << "Enter roll number: ";

cin >> rollNo;

cout << "Enter course: ";

cin >> course;

}

void displayStudentDetails() {

displayDetails(); // reuse base class function

cout << "Roll No: " << rollNo << ", Course: " << course << endl;

}

};

// Derived class: Teacher

class Teacher : public Person {

public:

string subject;

double salary;

void getTeacherDetails() {

getDetails(); // reuse base class function

cout << "Enter subject: ";

cin >> subject;

cout << "Enter salary: ";

cin >> salary;

}

void displayTeacherDetails() {

displayDetails(); // reuse base class function

cout << "Subject: " << subject << ", Salary: " << salary << endl;

}

};

int main() {

Student s;

Teacher t;

cout << "--- Enter Student Details ---" << endl;

s.getStudentDetails();

cout << "\n--- Student Information ---" << endl;

s.displayStudentDetails();

cout << "\n--- Enter Teacher Details ---" << endl;

t.getTeacherDetails();

cout << "\n--- Teacher Information ---" << endl;

t.displayTeacherDetails();

return 0;

}