**Module #3 Introduction to OOPS Programming**

**1. First C++ Program: Hello World**

**Program:**

#include <iostream> // Header for input/output

using namespace std;

int main() {

cout << "Hello, World!" << endl; // Output statement

return 0; // End of program

}

**Objective:**

* Understand basic structure:
  + #include <iostream> brings in the I/O functionality.
  + main() is the entry point of the program.
  + cout is used for output.

**2. Basic Input/Output**

**Program:**

#include <iostream>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

cin >> name; // Takes name (no spaces)

cout << "Enter your age: ";

cin >> age

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

return 0;

}

**Objective:**

* Practice with cin for input and cout for output.
* Understand how data flows between the user and the program.

**3. POP vs. OOP Comparison Program**

**A. POP (Procedural Programming) – *Area of a Rectangle***

#include <iostream>

using namespace std;

int main() {

float length, width, area;

cout << "Enter length: ";

cin >> length;

cout << "Enter width: ";

cin >> width;

area = length \* width;

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

return 0;

}

**B. OOP (Object-Oriented Programming) – *Area of a Rectangle***

#include <iostream>

using namespace std;

class Rectangle {

private:

float length, width;

public:

void input() {

cout << "Enter length: ";

cin >> length;

cout << "Enter width: ";

cin >> width;

}

float calculateArea() {

return length \* width;

}

void display() {

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

}

};

int main() {

Rectangle rect;

rect.input();

rect.display();

return 0;

}

**Objective:**

* POP focuses on functions and procedures.
* OOP uses classes and objects to encapsulate data and behavior.

**4. Setting Up Development Environment**

Before running this program, make sure your IDE (e.g., **Dev C++**, **Code::Blocks**, or **Visual Studio Code**) is installed and configured.

**Program:**

#include <iostream>

using namespace std;

int main() {

int num1, num2, sum;

cout << "Enter first number: ";

cin >> num1;

cout << "Enter second number: ";

cin >> num2;

sum = num1 + num2;

cout << "The sum is: " << sum << endl;

return 0;

}

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

**1. Variables and Constants**

**Program:**

#include <iostream>

using namespace std;

int main() {

// Variable declarations

int age = 20;

float height = 5.9;

char grade = 'A';

bool isStudent = true;

// Constant declaration

const double PI = 3.14159;

// Display values

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

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

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

cout << "Is Student: " << isStudent << endl;

cout << "Value of PI (constant): " << PI << endl;

// Performing operations

int nextYearAge = age + 1;

cout << "Next year age: " << nextYearAge << endl;

return 0;

}

**Objective:**

* Learn the difference between **variables (changeable)** and **constants (fixed)**.
* Explore multiple **data types**: int, float, char, bool, double.

**2. Type Conversion**

**Program:**

#include <iostream>

using namespace std;

int main() {

int intVal = 42;

float floatVal = 3.14;

// Implicit type conversion (int to float)

float result1 = intVal + floatVal;

// Explicit type conversion (float to int)

int result2 = intVal + (int)floatVal;

// C++ style casting

int result3 = intVal + static\_cast<int>(floatVal);

cout << "Implicit conversion result (int + float): " << result1 << endl;

cout << "Explicit conversion result (float cast to int): " << result2 << endl;

cout << "C++ static\_cast result: " << result3 << endl;

return 0;

}

**Objective:**

* Show **implicit conversion** (done automatically by the compiler).
* Show **explicit conversion** using **type casting**.

**3. Operator Demonstration**

**Program:**

#include <iostream>

using namespace std;

int main() {

int a = 10, b = 3;

// 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 << "\n--- Relational 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

cout << "\n--- Logical Operators ---" << endl;

bool x = true, y = false;

cout << "x && y: " << (x && y) << endl;

cout << "x || y: " << (x || y) << endl;

cout << "!x: " << (!x) << endl;

// Bitwise operators

cout << "\n--- Bitwise Operators ---" << endl;

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

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

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

cout << "~a = " << (~a) << endl;

cout << "a << 1 = " << (a << 1) << endl;

cout << "a >> 1 = " << (a >> 1) << endl;

return 0;

}

**Objective:**

* Practice all major C++ **operators**:
  + Arithmetic (+, -, \*, /, %)
  + Relational (==, !=, >, <, >=, <=)
  + Logical (&&, ||, !)
  + Bitwise (&, |, ^, ~, <<, >>)

**3. Control Flow Statements**

**1. Grade Calculator**

**Program:**

#include <iostream>

using namespace std;

int main() {

int marks;

cout << "Enter your marks (0 - 100): ";

cin >> marks;

if (marks >= 90)

cout << "Grade: A+" << endl;

else if (marks >= 80)

cout << "Grade: A" << endl;

else if (marks >= 70)

cout << "Grade: B" << endl;

else if (marks >= 60)

cout << "Grade: C" << endl;

else if (marks >= 50)

cout << "Grade: D" << endl;

else

cout << "Grade: F (Fail)" << endl;

return 0;

}

**Objective:**

* Learn to use **if-else** conditions to perform decision-making in programs.

**2. Number Guessing Game**

**Program:**

#include <iostream>

#include <cstdlib> // for rand()

#include <ctime> // for time()

using namespace std;

int main() {

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

int number = rand() % 100 + 1; // Random number between 1 and 100

int guess;

cout << "Guess a number between 1 and 100: ";

while (true) {

cin >> guess;

if (guess == number) {

cout << "Correct! You guessed the number!" << endl;

break;

} else if (guess > number) {

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

} else {

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

}

}

return 0;

}

**Objective:**

* Practice using **while loops** with **conditional logic**.

**3. Multiplication Table**

**Program:**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number to display its multiplication table: ";

cin >> number;

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

cout << number << " x " << i << " = " << (number \* i) << endl;

}

return 0;

}

**Objective:**

* Practice **for loops** for repeated tasks.
* Understand how to generate formatted output.

**4. Nested Control Structures (Star Triangle)**

**Program:**

#include <iostream>

using namespace std;

int main() {

int rows;

cout << "Enter the number of rows for the triangle: ";

cin >> rows;

for (int i = 1; i <= rows; i++) {

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

cout << "\* ";

}

cout << endl;

}

return 0;

}

**🔺 Output for rows = 4:**

markdown

CopyEdit

\*

\* \*

\* \* \*

\* \* \* \*

**5. Arrays and Strings**

**1. Array Sum and Average**

**Program:**

#include <iostream>

using namespace std;

int main() {

int arr[100], n, sum = 0;

float average;

cout << "Enter number of elements: ";

cin >> n;

cout << "Enter " << n << " elements:\n";

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

cin >> arr[i];

sum += arr[i];

}

average = (float)sum / n;

cout << "Sum = " << sum << endl;

cout << "Average = " << average << endl;

return 0;

}

**Objective:**

* Practice working with **arrays**, loops, and arithmetic.
* Learn how to compute **sum and average** of elements.

**2. Matrix Addition (2x2 Matrices)**

**Program:**

#include <iostream>

using namespace std;

int main() {

int mat1[2][2], mat2[2][2], result[2][2];

cout << "Enter elements of first 2x2 matrix:\n";

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

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

cin >> mat1[i][j];

cout << "Enter elements of second 2x2 matrix:\n";

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

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

cin >> mat2[i][j];

// Matrix addition

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

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

result[i][j] = mat1[i][j] + mat2[i][j];

// Display result

cout << "Resultant Matrix:\n";

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

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

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

cout << endl;

}

return 0;

}

**Objective:**

* Practice with **2D arrays** (multi-dimensional).
* Understand **element-wise matrix operations**.

**3. String Palindrome Check**

**Program:**

#include <iostream>

#include <string>

using namespace std;

int main() {

string str, reversed = "";

cout << "Enter a string: ";

cin >> str;

// Reverse the string

for (int i = str.length() - 1; i >= 0; i--)

reversed += str[i];

// Compare original and reversed

if (str == reversed)

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

else

cout << str << " is not a palindrome." << endl;

return 0;

}

**Example Input/Output:**

Input: madam

Output: madam is a palindrome.

**Objective:**

* Practice **string manipulation**.
* Learn how to reverse and compare strings.

**6. Introduction to Object-Oriented Programming**

**1. Class for a Simple Calculator**

**Program:**

#include <iostream>

using namespace std;

class Calculator {

public:

float add(float a, float b) {

return a + b;

}

float subtract(float a, float b) {

return a - b;

}

float multiply(float a, float b) {

return a \* b;

}

float divide(float a, float b) {

if (b != 0)

return a / b;

else {

cout << "Error: Division by zero!" << endl;

return 0;

}

}

};

int main() {

Calculator calc;

float x, y;

cout << "Enter two numbers: ";

cin >> x >> y;

cout << "Addition: " << calc.add(x, y) << endl;

cout << "Subtraction: " << calc.subtract(x, y) << endl;

cout << "Multiplication: " << calc.multiply(x, y) << endl;

cout << "Division: " << calc.divide(x, y) << endl;

return 0;

}

**Objective:**

* Learn how to define and use a **class** and its **member functions**.

**2. Class for Bank Account (Encapsulation)**

**Program:**

#include <iostream>

using namespace std;

class BankAccount {

private:

float balance;

public:

BankAccount() {

balance = 0;

}

void deposit(float amount) {

if (amount > 0) {

balance += amount;

cout << "Deposited: " << amount << endl;

} else {

cout << "Invalid deposit amount!" << endl;

}

}

void withdraw(float amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

cout << "Withdrawn: " << amount << endl;

} else {

cout << "Insufficient balance or invalid amount!" << endl;

}

}

float getBalance() {

return balance;

}

};

int main() {

BankAccount account;

account.deposit(1000);

account.withdraw(500);

cout << "Current Balance: " << account.getBalance() << endl;

return 0;

}

**Objective:**

* Understand **encapsulation** by using **private data** and **public methods**.

**3. Inheritance Example: Person, Student, Teacher**

**Program:**

#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 showDetails() {

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

}

};

// Derived class: Student

class Student : public Person {

public:

string course;

void getStudentDetails() {

getDetails();

cout << "Enter course: ";

cin >> course;

}

void showStudentDetails() {

showDetails();

cout << "Course: " << course << endl;

}

};

// Derived class: Teacher

class Teacher : public Person {

public:

string subject;

void getTeacherDetails() {

getDetails();

cout << "Enter subject: ";

cin >> subject;

}

void showTeacherDetails() {

showDetails();

cout << "Subject: " << subject << endl;

}

};

int main() {

Student s;

Teacher t;

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

s.getStudentDetails();

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

t.getTeacherDetails();

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

s.showStudentDetails();

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

t.showTeacherDetails();

return 0;

}

**Objective:**

* Understand **inheritance** and **code reuse** using base and derived classes.