**MODULE: 3.1 (C++ Basic)**

**· WAP to print “Hello World” using C++**



**· What is OOP? List OOP concepts**

**Ans:**oop means object oriented programing languagethe main purpose of OOP is to deal with real world entity using programing language.

**List of OOP concept:**classes,object,inheritance,encapsulation,abstra-ction and polymorphism.

**· What is the difference between OOP and POP?**

OPP : OPP follw bottom-up Approch.

POP : POP follw top-down Approch.

OPP : OPP is highly secure.

POP : POP is les secure.

OPP : it is deal with data.

POP : it is deal with algoritham.

OPP : it take more memory then POP

POP : it take less memory then OPP

OPP : they have acess specificaion like public,private & ptotected.

POP : there is no any acess specification.

**MODULE: 3.2 (Programing with C++)**

**· WAP to create simple calculator using class**



**· Define a class to represent a bank account. Include the following members:**

**1. Data Member:**

**-Name of the depositor**

**-Account Number**

**-Type of Account**

**-Balance amount in the account**

**2. Member Functions**

**-To assign values**

**-To deposited an amount**

**-To withdraw an amount after checking balance**

**-To display name and balance**



**· Write a program to find the multiplication values and the cubic values using inline function**

#include <iostream>

#include <string>

class BankAccount {

private:

std::string depositorName;

int accountNumber;

std::string accountType;

double balance;

public:

void assignValues(const std::string& name, int accNumber, const std::string& accType, double initialBalance) {

depositorName = name;

accountNumber = accNumber;

accountType = accType;

balance = initialBalance;

}

void deposit(double amount) {

balance += amount;

std::cout << "Amount " << amount << " deposited successfully.\n";

}

void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

std::cout << "Amount " << amount << " withdrawn successfully.\n";

} else {

std::cout << "Insufficient balance. Withdrawal failed.\n";

}

}

void display() {

std::cout << "Depositor Name: " << depositorName << std::endl;

std::cout << "Account Balance: " << balance << std::endl;

}

};

int main() {

BankAccount account;

account.assignValues("AKSHAY", 123456, "Savings", 5000.0);

account.display();

account.deposit(2000.0);

account.display();

account.withdraw(3000.0);

account.display();

account.withdraw(10000.0);

account.display();

}

**· Write a program of Addition, Subtraction, Division, Multiplication using constructor.**

#include <iostream>

class Calculator {

public:

float num1;

float num2;

public:

Calculator(float number1, float number2) {

num1 = number1;

num2 = number2;

}

float addition() {

return num1 + num2;

}

float subtraction() {

return num1 - num2;

}

float division() {

return num1 / num2;

}

float multiplication() {

return num1 \* num2;

}

};

int main() {

float number1, number2;

std::cout << "Enter two numbers: ";

std::cin >> number1 >> number2;

Calculator calculator(number1, number2);

std::cout << "Addition: " << calculator.addition() << std::endl;

std::cout << "Subtraction: " << calculator.subtraction() << std::endl;

std::cout << "Division: " << calculator.division() << std::endl;

std::cout << "Multiplication: " << calculator.multiplication() << std::endl;

}

**· Assume a class cricketer is declared. Declare a derived class batsman from cricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Displaydata. (Single Inheritance)**

#include<iostream>

#include<string>

using namespace std;

string name;

int age;

class cricketer

{

public :

cricketer()

{

cout<<"Enter Your Name:"<<endl;

getline(cin,name);

cout<<"Enter Your Age:"<<endl;

cin>>age;

cout<<"Welcome "<<name<<" to our cricket team"<<endl;

cout<<"Your age is:"<<age<<endl;

}

void batsman()

{

int matches,total\_runs,avrage\_runs,best\_perfomance;

cout<<"matches:"<<endl;

cin>>matches;

cout<<"total\_run:"<<endl;

cin>>total\_runs;

avrage\_runs = total\_runs / matches;

cout<<"best\_perfomance"<<endl;

cin>>best\_perfomance;

cout<<"matches "<<matches<<endl;

cout<<"total\_runs "<<total\_runs<<endl;

cout<<"avrage\_runs "<<avrage\_runs<<endl;

cout<<"best\_perfomance "<<best\_perfomance<<endl;

}

};

int main()

{

cricketer c;

c.batsman();

}

**· Create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data.Write also Main function (Multiple Inheritance)**



**· Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students. (Multilevel Inheritance)**

#include<iostream>

using namespace std;

class students {

public:

void rollNumber() {

cout << "Roll number" << endl;

}

};

class test : public students {

public:

void subject1Marks() {

cout << "Subject 1 marks" << endl;

}

void subject2Marks() {

cout << "Subject 2 marks" << endl;

}

};

class result : public test {

public:

int calculateTotalMarks() {

return subject1Marks + subject2Marks;

}

void displayResult() {

rollNumber();

subject1Marks();

subject2Marks();

cout << "Result" << endl;

}

};

int main() {

result student;

student.rollNumber();

student.subject1Marks();

student.subject2Marks();

student.displayResult();

}

**· Write a program to Mathematic operation like Addition, Subtraction,Multiplication, Division Of two number using different parameters and Function Overloading**

#include<iostream>

using namespace std;

class Calculator

{

public:

int add(int a, int b)

{

return a + b;

}

int subtract(int a, int b)

{

return a - b;

}

int multiply(int a, int b)

{

return a \* b;

}

int divide(int a, int b)

{

if (b != 0)

return a / b;

else {

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

return 0;

}

}

};

int main()

{

Calculator calc;

int sum = calc.add(5, 3);

cout << "Sum of 5 and 3 : " << sum << endl;

int diff = calc.subtract(8, 4);

cout << "Difference of 8 and 4: " << diff << endl;

int prod = calc.multiply(6, 4);

cout << "Product of 6 and 4 : " << prod << endl;

int quotient = calc.divide(10, 3);

cout << "Quotient of 10 divided by 3 : " << quotient << endl;

}

**· Write a Program of Two 1D Matrix Addition using Operator Overloading**

#include <iostream>

#include <vector>

class Matrix {

public:

std::vector<int> data;

int size;

public:

Matrix(int n) : size(n) {

data.resize(size);

}

int& operator[](int index) {

return data[index];

}

Matrix operator+(const Matrix& other) {

if (size != other.size) {

throw std::runtime\_error("Matrix sizes are not compatible for addition.");

}

Matrix result(size);

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

result[i] = data[i] + other.data[i];

}

return result;

}

};

int main() {

int n;

std::cout << "Enter the size of the matrices: ";

std::cin >> n;

Matrix matrix1(n);

Matrix matrix2(n);

std::cout << "Enter the elements of matrix1: ";

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

std::cin >> matrix1[i];

}

std::cout << "Enter the elements of matrix2: ";

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

std::cin >> matrix2[i];

}

Matrix sum = matrix1 + matrix2;

std::cout << "Sum of the matrices:" << std::endl;

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

std::cout << sum[i] << " ";

}

std::cout << std::endl;

}

**· Write a program to concatenate the two strings using Operator Overloading**

#include <iostream>

#include <cstring>

class MyString {

public:

char\* str;

int length;

public:

MyString(const char\* s = "") {

length = std::strlen(s);

str = new char[length + 1];

std::strcpy(str, s);

}

MyString(const MyString& other) {

length = other.length;

str = new char[length + 1];

std::strcpy(str, other.str);

}

~MyString() {

delete[] str;

}

MyString operator+(const MyString& other) {

int newLength = length + other.length;

char\* newStr = new char[newLength + 1];

std::strcpy(newStr, str);

std::strcat(newStr, other.str);

MyString result(newStr);

delete[] newStr;

return result;

}

friend std::ostream& operator<<(std::ostream& os, const MyString& myStr) {

os << myStr.str;

return os;

}

};

int main() {

MyString str1("Hello");

MyString str2(" World!");

MyString result = str1 + str2;

std::cout << "Concatenated string: " << result << std::endl;

}

**· Write a program to calculate the area of circle, rectangle and triangle using Function Overloading**



** Write a program to swap the two numbers using friend function without using third variable.**

#include <iostream>

using namespace std;

class Number {

public:

int value;

public:

Number(int num) {

value = num;

}

void display() {

cout << "Number: " << value << endl;

}

friend void swapNumbers(Number& num1, Number& num2);

};

void swapNumbers(Number& num1, Number& num2) {

num1.value = num1.value + num2.value;

num2.value = num1.value - num2.value;

num1.value = num1.value - num2.value;

}

int main() {

int num1, num2;

cout << "Enter number 1: ";

cin >> num1;

cout << "Enter number 2: ";

cin >> num2;

Number number1(num1);

Number number2(num2);

cout << "Before swapping:" << endl;

number1.display();

number2.display();

swapNumbers(number1, number2);

cout << "After swapping:" << endl;

number1.display();

number2.display();

}

** Write a program to find the max number from given two numbers using friend function.**

#include <iostream>

using namespace std;

class Number {

public:

int value;

public:

Number(int num) {

value = num;

}

void display() {

cout << "Number: " << value << endl;

}

friend int findMax(Number& num1, Number& num2);

};

int findMax(Number& num1, Number& num2) {

return (num1.value > num2.value) ? num1.value : num2.value;

}

int main() {

int num1, num2;

cout << "Enter number 1: ";

cin >> num1;

cout << "Enter number 2: ";

cin >> num2;

Number number1(num1);

Number number2(num2);

cout << "Number 1: " << number1.display();

cout << "Number 2: " << number2.display();

int maxNumber = findMax(number1, number2);

cout << "The maximum number is: " << maxNumber << endl;

}

**MODULE: 3.3 (C, C++ Templates)**

** Write a program of to swap the two values using templates**



** Write a program of to sort the array using templates.**

#include <iostream>

using namespace std;

template <typename T>

void sortArray(T arr[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

T temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

template <typename T>

void displayArray(T arr[], int size) {

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

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

int intArray[] = { 5, 3, 9, 1, 7 };

int intSize = sizeof(intArray) / sizeof(intArray[0]);

double doubleArray[] = { 3.14, 1.1, 7.2, 2.5, 5.8 };

int doubleSize = sizeof(doubleArray) / sizeof(doubleArray[0]);

char charArray[] = { 'z', 'x', 'a', 'b', 'c' };

int charSize = sizeof(charArray) / sizeof(charArray[0]);

cout << "Before sorting:" << endl;

cout << "Integer array: ";

displayArray(intArray, intSize);

cout << "Double array: ";

displayArray(doubleArray, doubleSize);

cout << "Character array: ";

displayArray(charArray, charSize);

sortArray(intArray, intSize);

sortArray(doubleArray, doubleSize);

sortArray(charArray, charSize);

cout << "After sorting:" << endl;

cout << "Integer array: ";

displayArray(intArray, intSize);

cout << "Double array: ";

displayArray(doubleArray, doubleSize);

cout << "Character array: ";

displayArray(charArray, charSize);

}