OOPS Assignment 6

Arnav Bansal

SID : 23103046

1. Write a C++ Program to demonstrate function overloading by creating a class "Math" with

overloaded functions "add" to add two integers, two floats, and two double values.

‘#include <bits/stdc++.h>

using namespace std;

class Math {

public:

int add(int a, int b) {

return a + b;

}

float add(float a, float b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

};

int main() {

Math m;

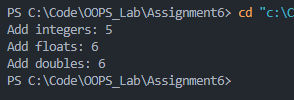
cout << "Add integers: " << m.add(2, 3) << endl;

cout << "Add floats: " << m.add(2.5f, 3.5f) << endl;

cout << "Add doubles: " << m.add(2.5, 3.5) << endl;

return 0;

}



2. Write a C++ Program that demonstrates the concept of function overloading by creating

overloaded functions "multiply" to multiply two integers, two floats, and two double values.

Use a class "Multiplier" to encapsulate these functions.

#include <bits/stdc++.h>

using namespace std;

class Multiplier {

public:

int multiply(int a, int b) {

return a \* b;

}

float multiply(float a, float b) {

return a \* b;

}

double multiply(double a, double b) {

return a \* b;

}

};

int main() {

Multiplier m;

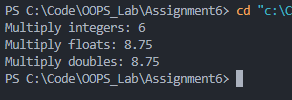
cout << "Multiply integers: " << m.multiply(2, 3) << endl;

cout << "Multiply floats: " << m.multiply(2.5f, 3.5f) << endl;

cout << "Multiply doubles: " << m.multiply(2.5, 3.5) << endl;

return 0;

}



3. Write a C++ Program that overloads a function "print" to print different types of data

(integer, float, and string). Create a class "Printer" that contains the overloaded "print"

Functions.

#include <bits/stdc++.h>

using namespace std;

class Printer {

public:

void print(int a) {

cout << "Integer: " << a << endl;

}

void print(float a) {

cout << "Float: " << a << endl;

}

void print(string a) {

cout << "String: " << a << endl;

}

};

int main() {

Printer p;

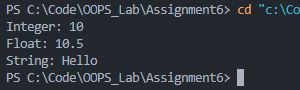
p.print(10);

p.print(10.5f);

p.print("Hello");

return 0;

}



4. Write a C++ Program to demonstrate constructor overloading by creating a class

Rectangle. Implement multiple constructors:

A default constructor that initializes the length and breadth to 1.

A parameterized constructor that takes two arguments to initialize the length and

breadth.

A constructor that takes one argument to initialize a square.

#include <bits/stdc++.h>

using namespace std;

class Rectangle {

public:

int length, breadth;

Rectangle() {

length = 1;

breadth = 1;

}

Rectangle(int l, int b) {

length = l;

breadth = b;

}

Rectangle(int side) {

length = side;

breadth = side;

}

};

int main() {

Rectangle r1;

Rectangle r2(3, 4);

Rectangle r3(5);

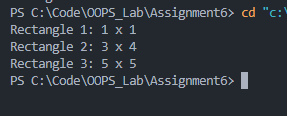
cout << "Rectangle 1: " << r1.length << " x " << r1.breadth << endl;

cout << "Rectangle 2: " << r2.length << " x " << r2.breadth << endl;

cout << "Rectangle 3: " << r3.length << " x " << r3.breadth << endl;

return 0;

}



5. Write a C++ Program to demonstrate constructor overloading in a class Time. Implement:

A default constructor that initializes hours, minutes, and seconds to 0.

A parameterized constructor that takes hours, minutes, and seconds as arguments.

Another constructor that takes the total seconds as an argument and converts it into

hours, minutes, and seconds.

#include <bits/stdc++.h>

using namespace std;

class Time {

public:

int hours, minutes, seconds;

Time() {

hours = 0;

minutes = 0;

seconds = 0;

}

Time(int h, int m, int s) {

hours = h;

minutes = m;

seconds = s;

}

Time(int totalSeconds) {

hours = totalSeconds / 3600;

minutes = (totalSeconds % 3600) / 60;

seconds = totalSeconds % 60;

}

};

int main() {

Time t1;

Time t2(2, 30, 45);

Time t3(3665);

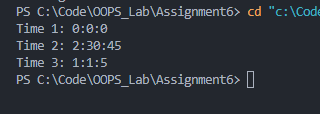
cout << "Time 1: " << t1.hours << ":" << t1.minutes << ":" << t1.seconds << endl;

cout << "Time 2: " << t2.hours << ":" << t2.minutes << ":" << t2.seconds << endl;

cout << "Time 3: " << t3.hours << ":" << t3.minutes << ":" << t3.seconds << endl;

return 0;

}



6. Write a C++ Program to overload the "+" operator to add two complex numbers. Define a

class "Complex" with real and imaginary parts as data members.

#include <bits/stdc++.h>

using namespace std;

class Complex {

public:

int real, imag;

Complex(int r = 0, int i = 0) : real(r), imag(i) {}

Complex operator+(const Complex& other) {

return Complex(real + other.real, imag + other.imag);

}

void display() {

cout << real << " + " << imag << "i" << endl;

}

};

int main() {

Complex c1(2, 3), c2(4, 5);

Complex c3 = c1 + c2;

cout << "Complex number 1: ";

c1.display();

cout << "Complex number 2: ";

c2.display();

cout << "Sum: ";

c3.display();

return 0;

}

