

**NAME:**

MUHAMMAD TALHA ZAHID

**SAPID:**

46287

**SEM/SEC:**

2-A

**SUBJECT:**

OBJECT ORIENTED PROGRAMMING

**INSTRUCTOR:**

SIR SHAHZAD

**DATE:**

12-05-2023

-------------------------------------------------------ASSIGNMENT#4-----------------------------------------------------------

## ### Part 1: Theory

**(Q1) Explain what polymorphism is and how it relates to object-oriented programming?**

**ANS:**

In object-oriented programming languages, polymorphism refers to having different forms or types of data that can be processed differently. Basically polymorphism means to add data to the program in various types and that can also processed in various forms in the program.

---------------------------------------------------------------------------------------------------------------------------------------

**(Q2) What is the difference between static and dynamic polymorphism?**

**ANS:**

Polymorphism in object-oriented programming can be static or dynamic, and their resolution at runtime differs.

* **Static :**

Static polymorphism, also known as compile-time polymorphism, refers to a type of polymorphism that is resolved by the compiler at compile-time. This type of polymorphism is achieved through method overloading, which allows a single class to have multiple methods with the same name but different parameters. Based on the number and type of arguments passed to the method, the compiler determines which method to call.

* **Dynamic**:

Dynamic polymorphism, also known as runtime polymorphism, refers to a type of polymorphism that is resolved at runtime. This type of polymorphism is achieved through method overriding, which allows a subclass to provide its own implementation of a method that is already defined in its parent class. It is the type of the object being referenced, rather than the type of the reference, that determines what method is called when the program is executed.

---------------------------------------------------------------------------------------------------------------------------------------

**(Q3) Describe the two types of polymorphism in C++.**

**ANS:**

In C++, there are two types of polymorphism: compile-time polymorphism and runtime polymorphism.

* **Compile-time polymorphism (static polymorphism):**

Compile-time polymorphism in C++ is achieved using function overloading and operator overloading. Function overloading allows multiple functions with the same name to coexist in a program as long as they have different parameter lists. Operator overloading enables operators such as +, -, \*, /, and << to be overloaded so that they can be used with custom data types.

compilation

* **Runtime polymorphism (dynamic polymorphism):**

Runtime polymorphism in C++ is achieved using virtual functions and inheritance. A virtual function is a function that is declared in a base class and can be overridden in a derived class. When a virtual function is called using a base class pointer or reference, the function that is executed is determined at runtime based on the actual object type that the pointer or reference is pointing to.

---------------------------------------------------------------------------------------------------------------------------------------

**(Q4) What is a virtual function? Explain why it is used.**

**ANS:**

In C++ programming language, a virtual function is a member function of a base class that can be overridden in a derived class. The virtual keyword is used to declare a function as virtual in the base class.

When a virtual function is called through a base class pointer or reference, the actual function that is executed is determined at runtime based on the type of the object that the pointer or reference is pointing to. This is known as dynamic dispatch or late binding.

The main purpose of using virtual functions is to enable runtime polymorphism in C++. By allowing derived classes to provide their own implementation of a virtual function, virtual functions make it possible to write code that can work with objects of different classes without knowing the exact type of each object at compile time. This makes the code more flexible, easier to maintain, and less error-prone.

---------------------------------------------------------------------------------------------------------------------------------------

**(Q5) Can a class have both virtual and non-virtual functions? Explain your answer.**

**ANS:**

Yes, a C++ class can have both virtual and non-virtual functions.

Virtual functions are used to achieve runtime polymorphism by allowing derived classes to provide their own implementation of a function. Non-virtual functions, on the other hand, are not meant to be overridden by derived classes and have a fixed implementation in the base class.

In some cases, it may make sense to have a mix of virtual and non-virtual functions in a class. For example, a base class may have some non-virtual functions that provide general behavior that is common to all derived classes, while also having virtual functions that can be overridden by specific derived classes to provide their own behavior.

---------------------------------------------------------------------------------------------------------------------------------------

## ### Part 2: Implementation

**(Q1) Write a C++ program that demonstrates the concept of function overloading.**

**ANS:**

#include <iostream>

using namespace std;

int add(int v6, int v7)

{

return v6;

}

int add(int v6, int v7, int v8)

{

return v6 + v8;

}

double add(double v6, double v7)

{

return v7 ;

}

int main()

{

int v1= 3, v2 = 4, v3 = 5;

double v4 = 1.2, v5 = 3.4;

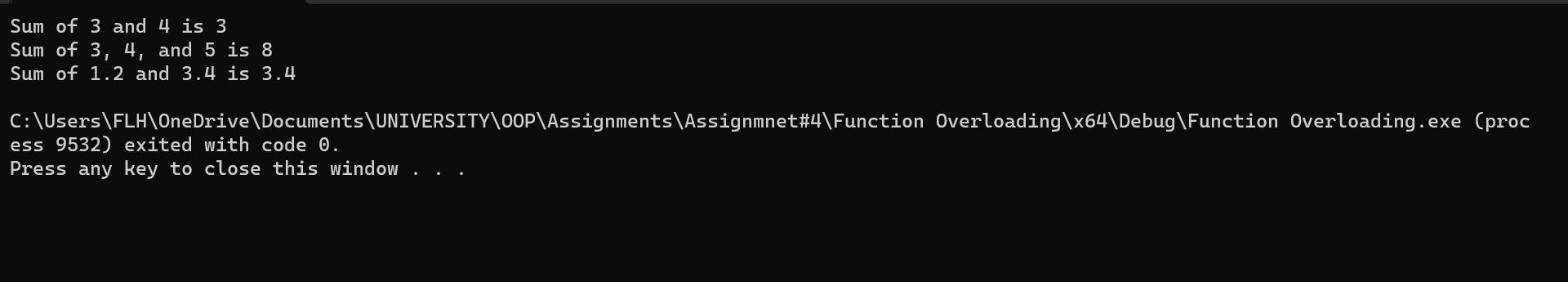
cout << "Sum of " << v1 << " and " << v2 << " is " << add(v1, v2) << endl;

cout << "Sum of " << v1 << ", " << v2 << ", and " << v3 << " is " << add(v1, v2, v3) << endl;

cout << "Sum of " << v4 << " and " << v5 << " is " << add(v4, v5) << endl;

return 0;

}



---------------------------------------------------------------------------------------------------------------------------------------

**(Q2) Write a C++ program that demonstrates the concept of operator overloading.**

**ANS:**

#include <iostream>

using namespace std;

class Complex

{

public:

double real;

double imag;

Complex() : real(0), imag(0) {}

Complex(double r, double i) : real(r), imag(i) {}

Complex operator+(const Complex& other) const

{

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

}

Complex operator-(const Complex& other) const

{

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

}

Complex operator\*(const Complex& other) const

{

double r = real \* other.real - imag \* other.imag;

double i = real \* other.imag + imag \* other.real;

return Complex(r, i);

}

friend ostream& operator<<(ostream& os, const Complex& c)

{

os << c.real << " + " << c.imag << "i";

return os;

}

};

int main()

{

Complex a(3, 4);

Complex b(1, 2);

Complex c = a + b;

Complex d = a - b;

Complex e = a \* b;

cout << "a = " << a << endl;

cout << "b = " << b << endl;

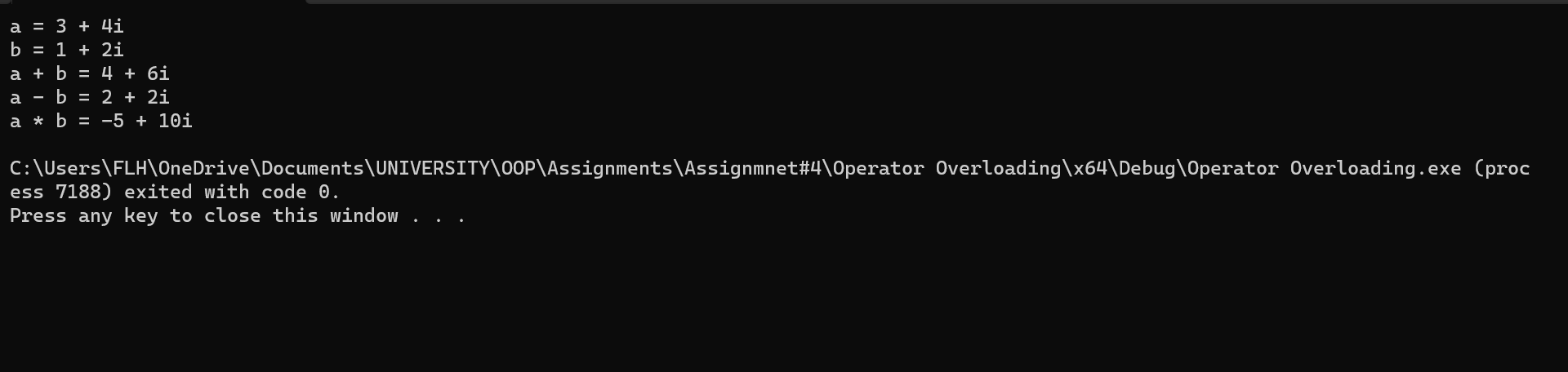
cout << "a + b = " << c << endl;

cout << "a - b = " << d << endl;

cout << "a \* b = " << e << endl;

return 0;

}



---------------------------------------------------------------------------------------------------------------------------------------

**(Q3) Write a C++ program that demonstrates the concept of runtime polymorphism using virtual functions.**

**ANS:**

#include <iostream>

using namespace std;

class Shape

{

public:

virtual double area()

{

cout << "This is the area method of the Shape class" << endl;

return 0;

}

};

class Rectangle : public Shape

{

private:

double length;

double width;

public:

Rectangle(double l = 0.0, double w = 0.0)

{

length = l;

width = w;

}

double area()

{

cout << "This is the area method of the Rectangle class" << endl;

return length \* width;

}

};

class Circle : public Shape

{

private:

double radius;

public:

Circle(double r = 0.0)

{

radius = r;

}

double area()

{

cout << "This is the area method of the Circle class" << endl;

return 3.14 \* radius \* radius;

}

};

int main() {

Shape\* s;

Rectangle r(3.0, 4.0);

Circle c(5.0);

s = &r;

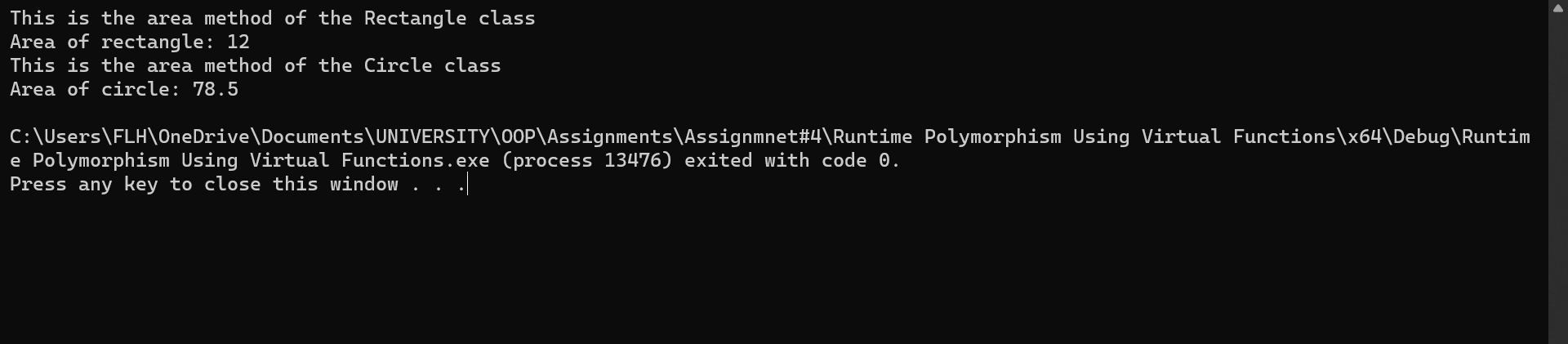
cout << "Area of rectangle: " << s->area() << endl;

s = &c;

cout << "Area of circle: " << s->area() << endl;

return 0;

}



---------------------------------------------------------------------------------------------------------------------------------------

**(Q4) Write a C++ program that demonstrates the concept of compile-time polymorphism using templates.**

**ANS:**

#include <iostream>

using namespace std;

template<typename T>

T add(T x, T y)

{

return x + y;

}

int main()

{

int a = 5, b = 10;

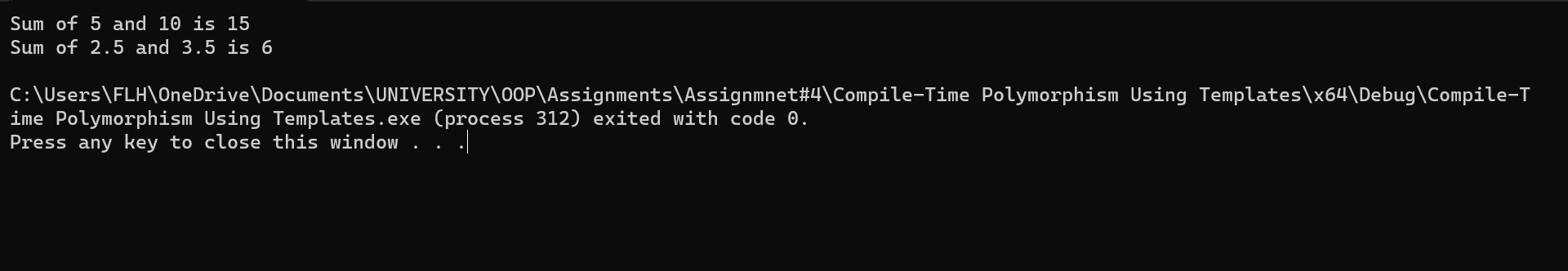
double c = 2.5, d = 3.5;

cout << "Sum of " << a << " and " << b << " is " << add(a, b) << endl;

cout << "Sum of " << c << " and " << d << " is " << add(c, d) << endl;

return 0;

}



---------------------------------------------------------------------------------------------------------------------------------------

## ### Part 3: Application

1. **Write a C++ program that uses polymorphism to create a hierarchy of shapes. The program should have a base class called `Shape` and derived classes for different types of shapes (e.g. `Circle`, `Rectangle`, `Triangle`). Each derived class should implement a function called `area()` that calculates the area of the shape. The program should allow the user to create objects of different shapes and calculate their areas using polymorphism.**

**Extend the previous program to include a function that sorts an array of shapes based on their area. The function should use polymorphism to determine the area of each shape and compare them. The program should allow the user to create an array of shapes of different types and sizes and sort them by area.**

**ANS:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Shape

{

public:

virtual double area() const = 0;

virtual ~Shape() {}

};

class Circle : public Shape

{

private:

double radius\_;

public:

Circle(double radius) : radius\_(radius) {}

double area() const override { return 3.14 \* radius\_ \* radius\_; }

};

class Rectangle : public Shape

{

private:

double width\_;

double height\_;

public:

Rectangle(double width, double height) : width\_(width), height\_(height) {}

double area() const override { return width\_ \* height\_; }

};

class Triangle : public Shape

{

private:

double base\_;

double height\_;

public:

Triangle(double base, double height) : base\_(base), height\_(height) {}

double area() const override { return 0.5 \* base\_ \* height\_; }

};

bool compare\_area(const Shape\* s1, const Shape\* s2)

{

return s1->area() < s2->area();

}

int main()

{

vector<Shape\*> shapes;

shapes.push\_back(new Circle(5));

shapes.push\_back(new Rectangle(3, 4));

shapes.push\_back(new Triangle(2, 5));

cout << "Areas of shapes:" << std::endl;

for (auto s : shapes)

{

cout << s->area() << endl;

}

sort(shapes.begin(), shapes.end(), compare\_area);

cout << "Sorted areas of shapes:" << endl;

for (auto s : shapes) {

cout << s->area() << endl;

}

for (auto s : shapes)

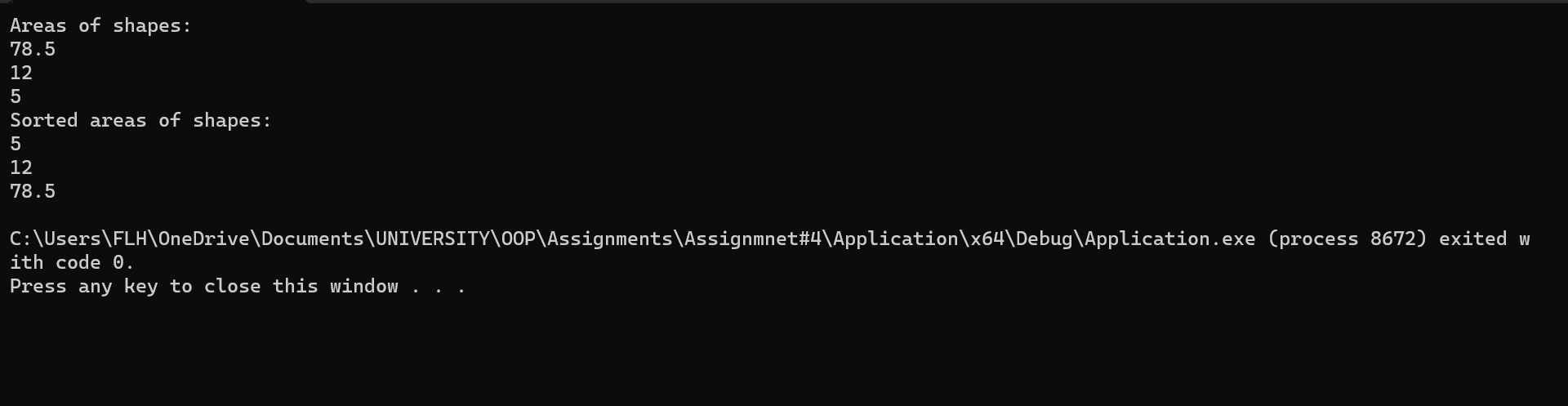
{

delete s;

}

return 0;

}



---------------------------------------------------------------------------------------------------------------------------------------

## ### Part 4: Reflection

**(Q1) Reflect on what you learned in this assignment. What was challenging, and what did you find interesting?**

**ANS:**

I have learnt the basic concepts of Polymorphism in this assignment. Yes, the task specially the application part#3 was a little bit challenging but if we analyze the statement correctly then I think it is not challenging task if we have been serious in class and labs.

---------------------------------------------------------------------------------------------------------------------------------------

**(Q2) How can you apply what you learned in this assignment to future projects or your future career?**

**ANS:**

I could easily apply what I have learned in this assignment at first I will apply the concept of Runtime Polymorphism Using Virtual Functions, Compile-Time Polymorphism Using Templates, Function Overloading, Operator Overloading and Polymorphism in my Object Oriented Programming Project and surely add in future projects as well

---------------------------------------------------------------------------------------------------------------------------------------