

ASSIGNMENT NO.04

Name: Zainab Amjad

Sap Id: 47596

Section: 2A

Program: BSCS

Subject: OOP

Submitted to: Sir.Shahzad

Part 1: Theory

1. Explain what polymorphism is and how it relates to OOP.

Ans. Polymorphism is the ability of any data to be processed in more than one form. The word itself indicates the meaning as poly means many and morphism means types. Polymorphism is one of the most important concepts of OOP. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class.

1. Differentiate between static polymorphism and dynamic polymorphism.

Ans. Static Polymorphism:

Static Polymorphism decides which method to execute during compile time. Method Overloading is an example of static polymorphism, and it is required to happens static polymorphism .Static Polymorphism achieved through static binding. Static Polymorphism happens in the same class. Object assignment is not required for static polymorphism. Inheritance not involved for static polymorphism.

Dynamic Polymorphism:

Dynamic Polymorphism decides which method to execute in runtime. Method Overriding is an example of dynamic polymorphism, and it is required to happens dynamic polymorphism. Polymorphism achieved through dynamic binding. Dynamic Polymorphism happens between different classes .It is required where a subclass object is assigned to super class object for dynamic polymorphism. Inheritance involved for dynamic polymorphism.

1. Describe two types of Polymorphism.

Ans. Overloading:

Two or more method having same name but different argument in same class. It happens at compile time .In this method call to its definition has happens at compile time.

Overriding:

Two or more method having the same name and same argument but in different class. It happens at runtime. In this method call to its definition happens at runtime.

1. What is virtual function? Explain why it is used.

Ans. A virtual function is a member function that you expect to be redefined in derived classes. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived classes version of the function.

1. Can a class have both virtual and non virtual functions?

Ans. Yes, a class can have both virtual and non-virtual member functions. A virtual function is a function that is declared with the keyword "virtual" in the class definition and is intended to be overridden by derived classes. When a virtual function is called on a derived class object through a pointer or reference to the base class, the appropriate derived class function is called based on the runtime type of the object. A non-virtual function is a function that is not declared with the "virtual" keyword in the class definition and is not intended to be overridden by derived classes. When a non-virtual function is called on a derived class object through a pointer or reference to the base class, the base class function is always called.

Example:

class Shape {

public:

virtual double area() const {

}

void setColor(const std::string& color) {

}

};

In this example, the area() function is virtual and intended to be overridden by derived classes, while the setColor() function is non-virtual and not intended to be overridden.

Part 2: Implementation

1. Write a C++ program that demonstrate the concept of function overloading.

Solution:

#include <iostream>

using namespace std;

void SumNum(int A, int B);

void SumNum(int A, int B, int C);

void SumNum(int A, int B, int C, int D);

int main()

{

SumNum(1,2);

SumNum(1,2,3);

SumNum(1,2,3,4);

return 0;

}

void SumNum(int A, int B)

{

cout<< endl << "SUMNUM is : "<< A+B;

}

void SumNum(int A, int B, int C)

{

cout<< endl << "SUMNUM is : "<< A+B+C;

}

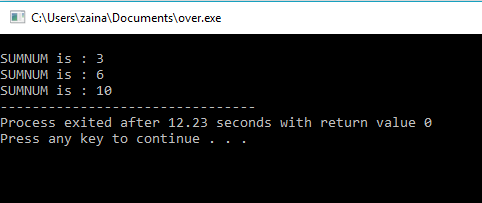
void SumNum(int A, int B, int C, int D)

{

cout<< endl << "SUMNUM is : "<< A+B+C+D;

}

OUTPUT:



1. Write a C++ program that demonstrate the concept of operator overloading.

Solution:

#include <iostream>

using namespace std;

class Frac {

private:

int a;

int b;

public:

Frac() : a(0), b(0) {}

void in() {

cout << "Enter the numerator : ";

cin >> a;

cout<< "Enter the denominator : ";

cin >> b;

}

Frac operator \* (const Frac &obj) {

Frac temp;

temp.a = a \* obj.a;

temp.b = b \* obj.b;

return temp;

}

void out() {

cout<<"The fraction is "<< a<<"/ "<<b;

}

};

int main() {

Frac F1, F2, result;

cout << "Enter the first fraction:\n";

F1.in();

cout << "Enter the second fraction:\n";

F2.in();

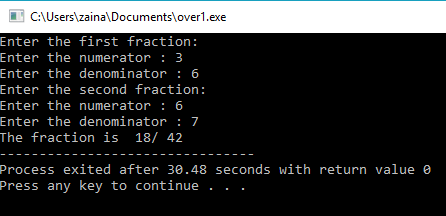
result = F1 \* F2;

result.out();

return 0;

}

OUTPUT:



1. Write a C++ program that demonstrate the concept of runtime polymorphism using virtual functions.

Solution:

#include <iostream>

using namespace std;

class Shape {

public:

virtual void calculate()

{

cout << "Area of your Shape ";

}

virtual ~Shape()

{

cout << "Shape Destructor Call\n";

}

};

class Rectangle : public Shape {

public:

int width, height, area;

void calculate()

{

cout << "Enter Width of Rectangle: ";

cin >> width;

cout << "Enter Height of Rectangle: ";

cin >> height;

area = height \* width;

cout << "Area of Rectangle: " << area << "\n";

}

virtual ~Rectangle()

{

cout << "Rectangle Destructor Call\n";

}

};

class Square : public Shape {

public:

int side, area;

void calculate()

{

cout << "Enter one side your of Square: ";

cin >> side;

area = side \* side;

cout << "Area of Square: " << area << "\n";

}

virtual ~Square()

{

cout << "Square Destructor Call\n";

}

};

int main()

{

Shape\* S;

Rectangle r;

S = &r;

S->calculate();

Square sq;

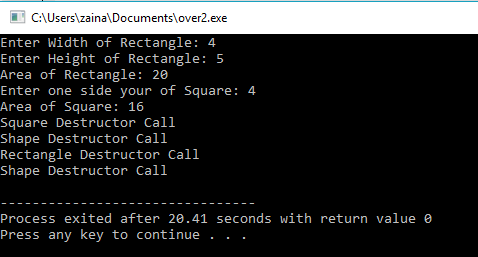
S = &sq;

S->calculate();

return 0;

}

OUTPUT:



1. Write a C++ program that demonstrate the concept of compile time polymorphism using templates.

Solution:

#include <iostream>

template <class T>

class StudentFee : T

{

public:

void annualFees()

{

this->fees();

}

};

class EngStudent

{

public:

void fees()

{

std::cout << "EngStudent Fees = RS. 70,000" << std::endl;

}

};

class CSStudent

{

public:

void fees()

{

std::cout << "CSStudent Fees = RS. 95,000" << std::endl;

}

};

int main()

{

StudentFee<EngStudent> engStudent;

StudentFee<CSStudent> csStudent;

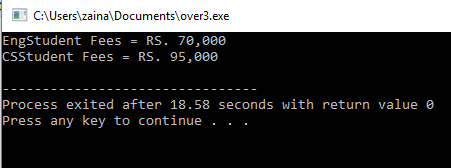
engStudent.annualFees();

csStudent.annualFees();

return 0;

}

OUTPUT:



Part 3: Application

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 a shape.The program should allow the user to create objects of different shapes and calculate their areas using polymorphism also 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

Solution:

#include <iostream>

using namespace std;

class Shape

{

protected:

double width, height;

public:

void set\_data (double a, double b)

{

width = a;

height = b;

}

virtual double area() = 0;

};

class Rectangle: public Shape

{

public:

double area ()

{

return (width \* height);

}

};

class Triangle: public Shape

{

public:

double area ()

{

return (width \* height)/2;

}

};

int main ()

{

Shape \*sPtr;

Rectangle Rect;

sPtr = &Rect;

sPtr -> set\_data (5,3);

cout << "Area of Rectangle is " << sPtr -> area() << endl;

Triangle Tri;

sPtr = &Tri;

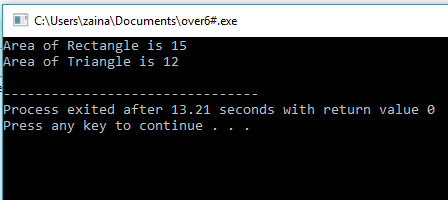
sPtr -> set\_data (4,6);

cout << "Area of Triangle is " << sPtr -> area() << endl;

return 0;

}

OUTPUT:



Part 4: Reflection

1. Reflect on what you learned in this assignment. What was the challenging, and what did you find interesting?

Ans. I learned a lot from this assignment. After doing this assignment, I understood polymorphism very well. When I was doing the application part, I was having a hard time with it.

1. How can you apply what you learned in this assignment to future projects or your future career?

Ans. Polymorphism is an important concept of OOP and it is very important for CS students to read all the concepts well then they will be able to work on big projects. After doing this assignment, I feel that my concept of polymorphism has been cleared very well.