PP Lab   Name: Shilpi Biswal
PRN: 22070122198

Practical No 9   **Date:** / /2022

Title: Create three classes: polygon (base class), rectangle and triangle (derived classes) having the same members: width, height, and functions set_values and area. Write a C++ program using run time polymorphism to implement it.

Description:
Polymorphism is a feature of OOP that allows the object to behave differently in different conditions
In C++ polymorphism is mainly divided into two types:
1) Compile time Polymorphism – This is also known as static (or early) binding
2) Runtime Polymorphism – This is also known as dynamic (or late) binding

Runtime Polymorphism is achieved using virtual functions.

Declare a pure virtual function inside the base class and redefine it in the derived classes
```
Class polygon
{
…
…
virtual void area()=0;      // pure virtual function
}
Class rectangle: public polygon
{
…
…
}
Class triangle : public polygon
{
…
…
}
main()
{
Polygon is an abstract class, it can’t be instantiated. Only pointer object can be created for this class
….
…
}
```

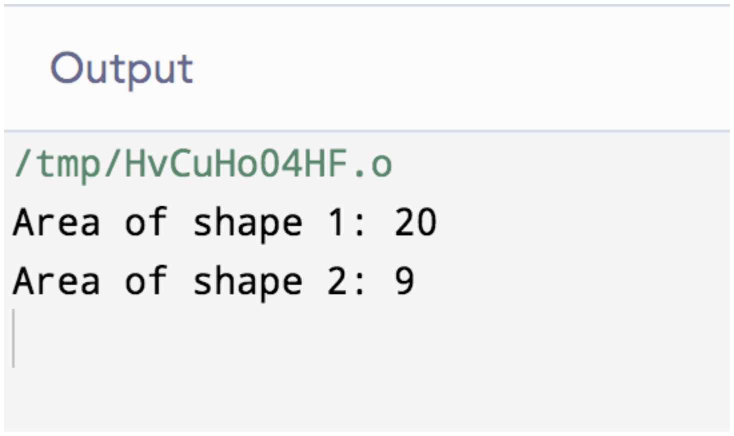
Program Code:
```
#include <iostream>
using namespace std;
```

PP Lab   Name: Shilpi Biswal
PRN: 22070122198

```
class Polygon {
protected:
   int width;
   int height;

public:
   void set_values(int w, int h) {
      width = w;
      height = h;
   }

   virtual int area() {
      return 0 ;// Base class area function, to be overridden in derived classes.
   }
};

class Rectangle : public Polygon {
public:
   int area() override {
      return width * height;
   }
};

class Triangle : public Polygon {
public:
   int area() override {
      return 0.5 * width * height;
   }
};

int main() {
   Polygon* shapes[2]; //  pointers to polygon objects

   Rectangle rect;
   Triangle tri;

   shapes[0] = &rect; // points to a Rectangle object
   shapes[1] = &tri;  // points to a Triangle object
```

```
    shapes[0]->set_values(4.0, 5.0);
    shapes[1]->set_values(3.0, 6.0);

    for (int i = 0; i < 2; i++) {
        cout << "Area of shape " << (i + 1) << ": " << shapes[i]->area() << std::endl;
    }

    return 0;
}
```

Input and Output



```
/tmp/HvCuHo04HF.o
Area of shape 1: 20
Area of shape 2: 9
```

Conclusion: Thus we have implemented the concept of run time polymorphism in C++.

Practice programs: Consider a book shop which sells both books and video tapes. Create a class media that stores the title and price of a publication. Derive two classes from media, one for storing the number of pages in a book and another for storing playing time of a tape. Write a C++ program using run time polymorphism to implement it.