Virtual function and runtime polymorphism

```cpp
// C++ program to demonstrate how we will calculate
// the area of shapes USING VIRTUAL FUNCTION
#include <fstream>
#include <iostream>
using namespace std;

// Declaration of Base class
class Shape {
public:
    // Usage of virtual constructor
    virtual void calculate()
    {
        cout << "Area of your Shape ";
    }
    // usage of virtual Destuctor to avoid memory leak
    virtual ~Shape()
    {
        cout << "Shape Destuctor Call\n";
    }
};

// Declaration of Derived class
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 Destuctor for every Derived class
    virtual ~Rectangle()
    {
```

```
        cout << "Rectangle Destuctor Call\n";
    }
};

// Declaration of 2nd derived class
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 Destuctor for every Derived class
    virtual ~Square()
    {
        cout << "Square Destuctor Call\n";
    }
};

int main()
{

    // base class pointer
    Shape* S;
    Rectangle r;

    // initialization of reference variable
    S = &r;

    // calling of Rectangle function
    S->calculate();
    Square sq;

    // initialization of reference variable
    S = &sq;

    // calling of Square function
    S->calculate();
```

```
    // return 0 to tell the program executed
    // successfully
    return 0;
}
```

Output

```
Enter Width of Rectangle: 10
Enter Height of Rectangle: 20
Area of Rectangle: 200
Enter one side your of Square: 16
Area of Square: 256
```