Aim:

To understand the concept of Virtual Function (polymorphism) by creating a base class `c_polygon` which has virtual function area and two derived classes `c_rectangle` and `c_triangle`. Calculate and return area for rectangle and triangle respectively.

Theory:

A virtual function a member function which is declared within base class and is re-defined (Overridden) by derived class. 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 class's version of the function. Some of the rules to define a virtual function are:-

i. They Must be declared in public section of class.
ii. Virtual functions cannot be static and also cannot be a friend function of another class.
iii. Virtual functions should be accessed using pointer or reference of base class type to achieve run time polymorphism.
iv. The prototype of virtual functions should be same in base as well as derived class.
v. They are always defined in base class and overridden in derived class. It is not mandatory for derived class to override (or re-define the virtual function), in that case base class version of function is
used.
vi. A class may have virtual destructor but it cannot have a virtual constructor.

Syntax :

```
class Base {
    public:
        // virtual function definition with keyword virtual
};
class Derived: public A {
    // polymorphism using function overloading
};
```

Code :

```
#include <iostream>
using namespace std;

class c_polygon{
    protected :
    float width, height;
    public :
    c_polygon() { }
    void setParams(float a, float b) {
        width = a;
        height = b;
    }
```

```
    virtual void getArea() {
        cout << "No shape selected" << endl;
    }
};

class c_rectangle : public c_polygon {
    public :
    c_rectangle() { }
    void getArea() {
        float area = width * height;
        cout << "\nLength : " << height << "  Breadth : " << width;
        cout << "\nArea of Rectangle : " << area << endl;
    }
};

class c_triangle : public c_polygon {
    public :
    c_triangle() { }
    void getArea() {
        float area = 0.5 * width * height;
        cout << "\nBase : " << width << "  Height : " << height;
        cout << "\nArea of Triangle : " << area << endl;
    }
};

int main(int argc, const char * argv[]) {
    // insert code here...
    c_polygon *ptr;
    c_rectangle rect;
    c_triangle tr;
    ptr = &rect;
    ptr->setParams(5,7);
    ptr->getArea();
    ptr = &tr;
    ptr->setParams(3,4);
    ptr->getArea();
    return 0;
}
```

Output :

```
PS D:\College\OOPS> .\virtual-function

Length : 7  Breadth : 5
Area of Rectangle : 35

Base : 3  Height : 4
Area of Triangle : 6
```

Discussion :

The above program illustrates the concept of virtual functions. In this program, a polygon class is made which has height and width as data members, a virtual function `getArea()` and function `setParams()` to accept parameters from user. Class `c_polygon` has two derived classes

`c_triangle` and `c_rectangle` . Both derived classes have function `getArea()` to calculate area and display. Thus virtual function polymorphism is demonstrated by the three classes.

Learning Outcomes :

- Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
- They are mainly used to achieve Runtime polymorphism.
- Functions are declared with a virtual keyword in base class.
- The resolving of function call is done at run-time.