Overriding

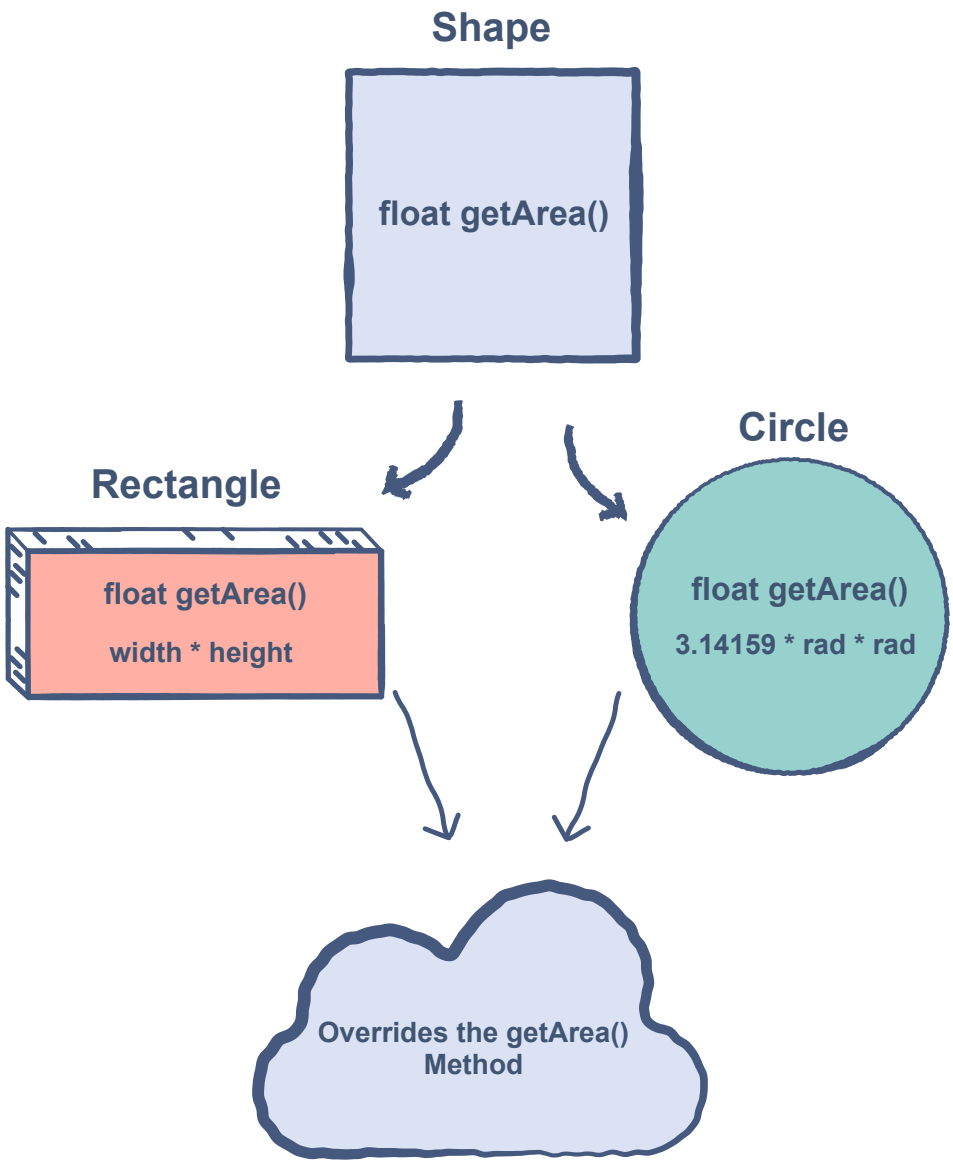
In this lesson, we'll be learning about how overriding is done in C++.

WE'LL COVER THE FOLLOWING

In object-oriented programming when we allow a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes is known as **Function Overriding**.

`getArea()` Overridden Function

As you have already seen the implementation of the function **getArea()** in the previous lesson, which depicts the concept of overriding.

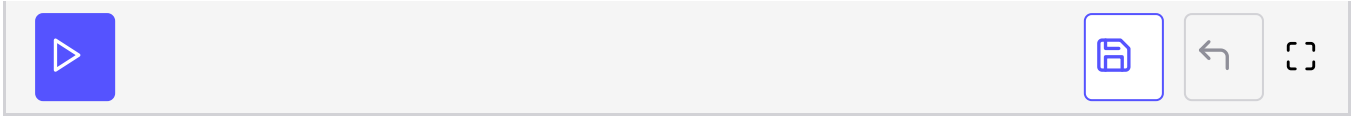


Implementation

Highlighted portion shows where overriding is done. Let's Have a look!

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

// A simple Shape interface which provides a method to get the Shape's area
class Shape {
  public:
  float getArea(){}
};

// A Rectangle is a Shape with a specific width and height
class Rectangle : public Shape {   // derived form Shape class
  private:
  float width;
  float height;

  public:
  Rectangle(float wid, float heigh) {
    width = wid;
    height = heigh;
```

```
  }
  float getArea(){
    return width * height;
  }
};

// A Circle is a Shape with a specific radius
class Circle : public Shape {
  private:
  float radius;

  public:
  Circle(float rad){
    radius = rad;
  }
  float getArea(){
    return 3.14159f * radius * radius;
  }
};

int main() {
  Rectangle r(2, 6);    // Creating Rectangle object

  Shape* shape = &r;   // Referencing Shape class to Rectangle object

  cout << "Calling Rectangle getArea function: " << r.getArea() << endl;      // Calls Rectar
  cout << "Calling Rectangle from shape pointer: " <<  shape->getArea() << endl <<endl; // Ca

  Circle c(5);    // Creating Circle object

  shape = &c;   // Referencing Shape class to Circle object

  cout << "Calling Circle getArea function: " << c.getArea() << endl;
  cout << "Calling Circle from shape pointer: " <<shape->getArea() << endl << endl;
}
```



Advantages of the Method Overriding

Method overriding is very useful in OOP and have many advantages. Some of them are stated below:

- The derived classes can give its own specific implementation to inherited methods without modifying the parent class methods.
- If a child class needs to use the parent class method, it can use it, and the other classes that want to have different implementation can use the overriding feature to make changes.

Key Features of Overriding

Here are some key features of the *Method Overriding:*

- Overriding needs inheritance and there should be at least one derived class.
- Derived class/es must have the same declaration, i.e., name, same parameters and same return type of the function as of the base class.
- The function in derived class/es must have different implementation from each other.
- The method in the base class must need to be overridden in the derived class.

In the next lesson, we'll learn about how to make a function virtual.