

Lesson 3:
Advanced OOP

SEARCH

RESOURCES

CONCEPTS

1. Polymorphism and Inheritance

2. Bjarne on Inheritance

3. Inheritance

4. Access Specifiers

5. Exercise: Animal Class

6. Composition

7. Exercise: Class Hierarchy

8. Exercise: Friends

9. Polymorphism: Overloading

10. Polymorphism: Operator Overlo...

11. Virtual Functions

12. Polymorphism: Overriding

13. Override

14. Multiple Inheritance

15. Generic Programming

16. Bjarne on Generic Programming

17. Templates

18. Bjarne on Templates

19. Exercise: Comparison Operation

20. Deduction

21. Exercise: Class Template

22. Summary

23. Bjarne on Best Practices with Cla...

Virtual Functions

SEND FEEDBACK

https://youtu.be/2krvZ3-INUk

Virtual Functions

Virtual functions are a polymorphic feature. These functions are declared (and possibly defined) in a base class, and can be overridden by derived classes.

This approach declares an **interface** at the base level, but delegates the implementation of the interface to the derived classes.

In this exercise, `class Shape` is the base class. Geometrical shapes possess both an area and a perimeter. `Area()` and `Perimeter()` should be virtual functions of the base class interface. Append `= 0` to each of these functions in order to declare them to be "pure" virtual functions.

A **pure virtual function** is a **virtual function** that the base class **declares** but does not **define**.

A pure virtual function has the side effect of making its class **abstract**. This means that the class cannot be instantiated. Instead, only classes that derive from the abstract class and override the pure virtual function can be instantiated.

```
class Shape {
public:
    Shape() {}
    virtual double Area() const = 0;
    virtual double Perimeter() const = 0;
};
```

Virtual functions can be defined by derived classes, but this is not required. However, if we mark the virtual function with `= 0` in the base class, then we are declaring the function to be a pure virtual function. This means that the base class does not define this function. A derived class must define this function, or else the derived class will be abstract.

Instructions

1. Create base class called `Shape`.

2. Define pure virtual functions (`= 0`) for the base class.

3. Write the derived classes.

- Inherit from `class Shape`.
- Override the pure virtual functions from the base class.

4. Test in `main()`

Saving Graffiti Recording. Please wait...

Menu

Expand

NEXT