

◀ 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 Overl...

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...

Polymorphism: Operator Overloading

SEND FEEDBACK

https://youtu.be/ejJ8uoPtFoo

Operator Overloading

. In this exercise you'll see how to achieve polymorphism with **operator overloading**. You can choose any operator from the ASCII table and give it your own set of rules!

Operator overloading can be useful for many things. Consider the `+` operator. We can use it to add `int`'s, `double`'s, `float`'s, or even `std::string`'s.

In order to overload an operator, use the `operator` keyword in the function signature:

```
Complex operator+(const Complex& addend) {  
    //...logic to add complex numbers  
}
```

Imagine vector addition. You might want to perform vector addition on a pair of points to add their x and y components. The compiler won't recognize this type of operation on its own, because this data is user defined. However, you can overload the `+` operator so it performs the action that you want to implement.

Instructions

1. Define class `Point`.

2. Declare a prototype of overload method for `+` operator.

3. Confirm the tests pass.

Saving Graftii Recording. Please wait...

↑ Menu

↗ Expand

NEXT