CS107: C++ PROGRAMMING

Back to 'Study Guides and Review Exercises'

Unit 3 Study Guide and Review: Object-Oriented Programming

3a. Define and compare/contrast constructors and deconstructors

1. What is object-oriented programming?
2. How do you create a constructor and when is it deployed?
3. Explain what it means to create an overloaded constructor?
4. How do you create a destructor and when is it deployed?

An explanation of object-oriented programming and the concepts that accompany it, including encapsulation, inheritance, abstraction, and polymorphism are explained in these lecture notes. Object-oriented programming allows a programmer to manage the complexity of a program by grouping code into self-contained boxes (classes). Constructors are a way to “call” an instance of a class. You define the class within a class file, and then create an instance of the class, usually within the main program, although it may be created within other methods. The creation of constructors and destructors can be reviewed if you watch this video.

3b. Create overloading operators

1. What is an overloaded operator and why is it used?

Overloaded operators allow you to redefine the meaning and purpose of an operator. Imagine a world where == will have a different behavior depending on the values that are sent to it. Although you may have the capability to overload operators, it is important that you use this sparingly, as it tends to create ambiguity in your program, as operators are not behaving as they are expected. To learn more, explore this article

3c. Define and use the keyword “this” and use the static members appropriately

1. What is the keyword “this” used for?

“this” is a keyword that is used to indicate the specific instance of an object. It is required to be used when the private data members have the same name as the value being passed. For example, you would use:

this.name = name; 'this' indicates to use this particular instance of the class. This video explores the use of the keyword "this".

1. When are static members used?

Static members are used when multiple instances of a class are going to need access to the same data values, and this article explores the use of static members. For example, if you create a data member as static, every instance of the class will refer to the same value in member, instead of having their own variable value. An example of when you might want to use this is if you are wanting to keep a count of the number of instances of the class that occurs within a program.

3d. Design and appropriately use friend functions and classes

1. What are friend functions and classes?

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions. A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends. For more information on friends, review this article.

3e. Use the class inheritance to design better code

1. What is inheritance and how is it implemented?

The idea of inheritance is that you identify those characteristics of an object that are common among all types of that object and create a class that shares these characteristics (the parent class). Then you create individual classes (child classes) to represent each specific type that identifies those characteristics that are unique to this type. These individual classes inherit the characteristics of the parent class. To learn more about inheritance, review these slides.

3f. Explain how polymorphism is achieved through C++ code

1. What is polymorphism?
2. How is polymorphism implemented?

Polymorphism is a way to implement multiple versions of the same method that are part of difference class definitions. When a child encounters a method that is part of their own class and a part of the parent class, the child class will implement their version of the method within their own class. To learn more information on this, review this article.

Unit 3 Vocabulary

This vocabulary list includes terms that might help you with the review items above and some terms you should be familiar with to be successful in completing the final exam for the course.

Try to think of the reason why each term is included.

- Abstraction
- Class
- Constructor
- Destructor
- Encapsulation

- Friend function
- Inheritance
- Object-oriented programming
- Overloading
- Polymorphism
- Static members
- This

Last modified: Wednesday, December 7, 2016, 2:28 PM