Inheritance in C++

Bryce Boe

2012/08/28

CS32, Summer 2012 B

Overview

- Assignment Operator
- Inheritance
  - Descendants and Ancestors
  - Instance variables and methods
  - Protected
  - Constructors
  - Calling ancestor functions
  - Polymorphism

Overloading Assignment Operator

- Used when assigning an object to another object:
  - Tuple a_tuple, b_tuple(5);
  - a_tuple = b_tuple;

void operator =(const Tuple &other);

Tuple& operator =(const Tuple &other);

Inheritance

- Classes can extend existing classes
- Member variables and member functions are inherited by the extending class
- An inheriting class is called a child class, derived class, or subclass
- The inherited calss is called a parent class, base class, or superclass.

How to inherit

```
class NamedTuple: public Tuple {
…
};
```

Ancestors and Descendants

- Any class that inherits directly or indirectly from some class is said to be a descendant of that class
  - C inherits B which inherits A
    - Both C and B are descendants of A
- Any class which has descendants, is said to be an ancestor of those classes
  - C inherits B which inherits A
    - A is an ancestor to both B and C

Constructors are not inherited

- Constructors need to be redefined (you probably want to anyway as you'll likely add instance variables to a new class)

Exception: If no constructors are defined in the child-class, then it will inherit the default constructor

Instance Methods and Variables

- Child classes inherit both instance methods and variables
- Private instance methods are unusable as they cannot be invoked
- Private instance variables still store data, but they can only be used via accessors and mutators

Protected Keyword

- Declaring variables as protected in the parent class, allows descendant classes to access them directly

Redefining Instance Methods

- Instance methods can be redefined in child classes
- The method just needs to be declared and defined in the context of the child class

Note: Function names only have to be declared if new or being redefined

Destructor

- The parent class destructors are automatically invoked after the child class destructors

virtual keyword

- The virtual keyword allows for dynamic dispatch
- Dynamic dispatch means to lookup the appropriate function to call at run-time
- Useful when working only with ancestor class types, but you want to call the specific descendant class function

More on Virtual

- The virtual property is inherited, thus it isn't required to have virtual in the function declaration when overriding a method

Polymorphism

- The use of the virtual keyword is polymorphism
- The behavior of an instance changes depending on its actual implementation