More Classes in C++

Bryce Boe

2012/08/20

CS32, Summer 2012 B

Overview

- Reminder: Midterm Wednesday
- Project 1 Part 1 Solution
- Project 1 Part 2 Overview
- Classes Recap
- More on C++ Classes

Project 1 Part 1 Solution

- Example solution presented in class
  - Will not be available online

Project 1 Part 2

- Compare function
- Using test_it.sh to test sorter
  - Add test files
  - Run via:
    - PATH=$PATH:. test_it.sh tests/

Classes

- Provide encapsulation
  - Combining a number of items, such as variables and functions, into a single package, such as an object of some class (or instance of the class)
- Today: Discuss encapsulating functionality provided by the programming language

Overloading Instance Methods

- Defining methods of a class with the same name, but different parameters

```
void Date::update_date(int d, int m, int y) {…}
void Date::update_date(Date &other) {…}
```

- Today: Look at overloading operators (+, -, <<, >>, =, ==, etc.)

More C++ Classes

- friend functions
- const keyword
- overloading operators
- arrays in classes
- destructor
- copy constructor
- overloading assignment operator

friend functions

- A friend function is a function that has access to the private members of a class
- The function must be declared inside the class

```
class Date {
friend bool equals(const Date &, const Date&);
…
};
```

const keyword

bool Date::equal(const Date& other) const{…}

- const for parameters
  - Means the method cannot modify the parameter
- const at the end of the function declaration
  - Means that the method cannot modify its own state

Instance method or not?

- (In book) member v. non-member function
- Book's suggestion:
  - Use member functions if operating on a single instance
  - Use non-member friend functions otherwise
- It's up to you to choose:

```
friend bool equals(const Date &, const Date&);
bool equals(const Date &) const;
```

Overloading Operators

- Special syntax using operator keyword

- Add an int on the rhs (Date() + 5)

```
Date operator +(int other) const;  // member
friend Date operator +(const Date &lhs, int rhs);
```

- Add an int on the lhs (5 + Date())

```
friend Date operator +(int lhs, const Date &rhs);
```

Overloading << for output

- In the following example, the left hand sides are always of type *ostream&*.
  - cout << date_a << " " << date_b
  - Parsed as ((cout << date_a) << " ") << date_b
- Function prototype for Date objects

friend ostream& operator <<(ostream &stream,
const Date &date);

Overloading >> for input

- Similar behavior to output

```
friend istream& operator >>(istream &stream,
                           Date &date);
```

Arrays in classes

- You can have arrays of classes
  - Date some_dates[10];
- You can have arrays in your classes
  - Static arrays
    - Memory is cleaned up automatically just as all other instance variables when the object goes out of scope
  - Dynamic arrays (and other objects created with new)
    - Need a special way to clean these up when the object goes out of scope

Destructor

- An instance's destructor is called
  - implicitly when the instance goes out of scope
  - explicitly when delete is called on the instance
- Needs to be declared whenever dynamic memory is used in the class
- Can also be used for class "clean up"

~ClassName();

Copy constructor

- Default copy constructor will only perform a shallow copy
  - Pointers will point to the same dynamic data
- Copy constructor should implement a deep copy
- Used when returning an instance of the object, or passing the object by value

```
Tuple(const Tuple &other);
```

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);

The Big Three

- Copy constructor
- Assignment operator
- Destructor

- If you have to implement any one of them, then you should implement all of them

For Tomorrow

- Review all material up to today
- Come to class with questions
- (Recommended) implement at least 1 of the sort functions