Classes in C++

Bryce Boe

2012/08/15

CS32, Summer 2012 B

Overview

- Finish Sorting recap
- Thinking object oriented recap
- Classes in C++
- Building a class in C++ (real time demo)

Sorting recap

- ~~Bubble sort~~
- ~~Insertion sort~~
- ~~Selection sort~~
- Merge sort
- Heapsort
- Quicksort

Thinking object oriented recap

- Language as an influence of thought process
- OO concepts
  - Separation of interface and implementation
  - Information hiding
  - Inheritance
- Writing reusable code

Exciting Note for Today

- The gcc compiler now requires C++ to build
  - Essentially means parts of the gcc compiler are written in C++

- http://gcc.gnu.org/git/?p=gcc.git;a=commit;h=2b15d2ba7eb3a25dfb15a7300f4ee7a141ee8539

Structures

- Structures provide a way to organize data
- Structures in C++ are essentially classes, not true in C

Classes

- An object is a variable that has member functions (instance methods)
- A class is a data type whose variables are objects
- Class
  - Describe the kind of values the variables hold (state)
  - Describe the member functions (behavior)

Terminology

- The book uses member to mean a particular instance of a *class*
- The book uses members to mean attributes of a class (variables and methods)
- Function and method are somewhat used interchangeably
- Similar:
  - member variable = instance variable
  - member method = instance method

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)

Scope Resolution Operator

- ClassName::method_name
- Used to identify the scope, class in this case, that the method belongs to as there may be more than 1 instance of method_name
- Scope resolution isn't necessary if you are also a member of that class

Data Hiding

- Declaring member (instance) variables as private, why?
  - Assists in separation of implementation and interface
  - Allows for input validation and state consistency

Declaring Private attributes

```
class Date {
    int day;        // this section is private by default
    int month;  // though you should be explicit
public:
    void output_date();
private:
    int year;
};
```

Accessor methods

- Sometimes called getters
- Instance methods that return some data to indicate the state of the instance
- Typically prefixed with get_

```
int Date::get_day() { return day; }
```

Mutator methods

- Sometimes called setters
- Instance methods that update or modify the state of the instance
- Typically prefixed with set_

```
void Date::set_day(int d) { day = d; }
```

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) {…}
```

Class Constructors

- A constructor is used to initialize an object
- It must:
  - Have the same name as the class
  - Not return a value
- Constructors should be declared public
  - To ponder: what does it mean to have a non-public constructor?
- Always define a default constructor

Example

```
class Date {
  public:
    Date(int d, int m, int y);
    Date();  // default constructor
  private:
    int day, month, year;
};
```

Two ways to initialize variables

- From the constructor declaration (implementation)
- Method 1: Initialize in the constructor initialization section

```
Date::Date() : day(0), month(0), year(0) {}
```

- Method 2: In the method body

```
Date::Date() {
   day = 0; month = 0; year = 0; }
```

Example Constructor Usage

```
Date a (10, 10, 11);  // use the 3 param constructor
Date b;  // correct use of default constructor

~~Date c();~~ // incorrect use of default constructor
              // This is actually a function definition

Date d = Date(); // valid, but inefficient
```

Anonymous Instances

- An instance that is not bound to a variable

Date d = Date();

- In the above example there are actually two instances of class Date
  - The first is represented by d
  - The second is the anonymous instance represented by Date()
- The assignment operator is used to transfer information from the anonymous instance to d

Abstract Data Types

- A formal specification of the separation of implementation and interface
- Developer can use ADTs without concern for their implementation
- Using classes, you can define your own ADTs
  - This allows for reusable code

Tips for writing ADTs

- Make all the member variables private attributes of the class
- Provide a well defined public interface to the class and **don't** change it
- Make all helper functions private

Intro to Inheritance in C++

- Derived (aka child or sub) classes take on (inherit) the attributes of the parent (aka base or super) class

```
class Timestamp : public Date {
…
};
```

For Lab2

- Read "The *const* Parameter Modifier" section
  - Page 620 in the textbook

int Date::days_until(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 not modify its own instance's state

For Monday

- Read chapter 11 in the C++ book
  - Again, think about OO design themes in the C++ context
- The textbook is available in the library

Building a class demo