More C++

Bryce Boe

2013/07/18

CS24, Summer 2013 C

Outline

- Project 1 Review
- Finish C++ Introduction

PROJECT 1 REVIEW

Sample Solution

- <In class explanation of sample code>
  - Solution will NOT be posted

C++ INTRODUCTION CONTINUED

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

Finish Example from Tuesday

- <In class completion of converting struct Date to class Date (encapsulation.cpp)>