Exam 2 Review

CS 2308
Spring 2013

Jill Seaman

Exam 2

- Wed, Apr. 10 and Thurs, Apr. 11
- In class, closed book, closed notes, clean desk
- 20% of your final grade
- 80 minutes to complete it
- I recommend using a pencil (and eraser)
- All writing will be done on the test paper I will hand out. (I will bring extra paper).
- No calculators.

Exam Format

- 100 points total
  - Writing programs/functions/code
  - Multiple choice (10 at 2pts each)
  - Fill-in-the-blank/short answer
  - Tracing code (what is the output)

Which Lectures?

- Chapter 13
- Multi-file Development
- Chapter 14
- Pointers and this
- Chapter 17 (slides 1-20)

Ch.13+14: Classes

- Procedural programming
  - What it is
  - What kind of problems (handling change to impl)
- Object oriented programming:
  - What it is
  - How it solves problems of procedural problems
  - **encapsulation**
  - **data (or information) hiding**
  - **interface**
  - class vs instance (object)

Ch.13+14: Classes (cont)

- Declaring a class:
  - Members: variables and functions
  - private vs public, access rules
  - syntax: class declaration
  - syntax: member function definitions
  - accessor/mutator (getter/setter)
  - How to define instances
  - How to access members
- Separating specifications from implementation
  - What goes where (class decl, member func defs)
  - How to compile

Ch.13+14: Classes (cont)

- Inline member functions
- Constructors
  - How to name, return type?
  - When are they called? what do they do?
  - Default constructor
  - passing args to constructors
- Destructors
  - what it is, how to name, when is it called?
- Overloaded constructors and member functions

Ch.13+14: Classes (cont)

- Arrays of objects: declare, initialize, and access
  - Write functions to process arrays of objects
- Instance vs Static Members: variables+functions
- Copy Constructor
  - Default copy constructor (member-wise assignment)
  - When+how to define your own
  - When is it used? (initialization, not assignment)
- Operator overloading
  - syntax of function definition, syntax of use
  - how to overload +, -, =, relational operators

11.9, 13.3: Pointers to Struct, Object and 14.5: the `this` pointer

- How to declare, assign pointers
- Structure pointer operator: ->
- How to access members through the pointer
  - `(*p).member vs. p->member vs. *(p.member)`
- dynamic allocation of structures, objects, arrays
- When an object is deleted, destructor is called
- the "this" pointer
  - what it points to
  - how it is used in member functions

C++ Programming on Linux

- Programming with multiple files (multi-file dev)
  - How to split up a program with classes
  - Header files vs .cpp files, when to include
  - How to compile multiple file program:
    - using g++ :
      $ g++ a.cpp b.cpp
    - separate compilation:
      $ g ++ -c a.cpp
      $ g++ -c b.cpp
      $ g++ a.o b.o
    - makefile:
      understand the example, basic rules, how to make

Ch.17: Linked Lists
(slides 1-20)

- Dynamically allocated list data structure
- Organization: nodes, head pointer, empty list
- Declaring a linked list datatype
  (Node structure and head pointer)

- creating an empty list
- appendNode
- finding the last node
- traversing a linked list (displayNode)

Example Programming Problem

Consider a Rectangle class, which stores a rectangle using two floating point values: width and height.

It has 2 constructors (one default, one with 2 arguments), and accessors and mutators for the width and height. It also has an overloaded < operator (a rectangle is less than another if its area is less than the other's area).

The default rectangle has width and length equal to 1. The other constructor takes the initial width and height. The rectangle also has a function area() that calculates the area of the rectangle.

(a) Write the class declaration.

(b) Implement all the class functions.

Example Tracing Problem

A)What will the EXACT output of the following program be?

```
class IntCell {
   private:
      int *storedValue;
   public:
      IntCell (int initialValue)
      { storedValue = new int;
        *storedValue = initialValue; }
      ~IntCell()
      {  delete storedValue;  }
      int read () const
      {  return *storedValue; }
      void write (int x)
      {  *storedValue = x;  }
};

void main() {
   IntCell object1(10);
   IntCell object2 = object1;
   object2.write(4);
   cout << object1.read() << endl;
   cout << object2.read() << endl;
}
```

B) Define a copy constructor for the IntCell class so that two instances of IntCell will not point to the same storedValue.

How to Study

- Review the slides
  - understand all the concepts
- Use the book to help understand the slides
  - there will be no questions over material (or code) that is in the book but not on the slides
- Review assignments + solutions
- Try some exercises from the book
- Practice, practice, practice
- Get some sleep