LECTURE 34: THE BIG THREE

Today

- Reading
  - Weiss Ch. 4.6
- Objectives
  - Recap lessons from Wednesday's lab!
  - The Big Three
  - Access modifiers
  - Exceptions in C++

Lessons from Wednesday's Lab

- Shallow copying vs. deep copying
- The default copy constructor and operator= provide a shallow copy!
  - They copy the instance variables but not any memory pointed to by the instance variables
  - Consequences include memory leaks and (possible) runtime errors
- *If an instance variable is a pointer to heap-allocated memory, need to overwrite the big three*

Classes in C++: The Big Three

- The Big Three
  - Destructor
  - Copy constructor
  - `operator=` function
- *Rule of Three: If you need to overwrite one of these, overwrite them all*
- To disallow a default copy constructor and operator=, declare private ones that do nothing

Access Modifiers in C++

- Java
  - public, protected, private, and "default"
- In C++
  - public – everybody
  - private – class only
  - protected – subclasses only
  - (friend – friends can access private section of a class)

Access Modifiers in C++

```
class Node{

    private:
        int element;
        Node *next;

        // the constructor is private!
        Node(int theElement, Node* n)
            : element(theElement), next(n){ }


        // the integer queue class is the only
        // class that can create Node variables
        friend class IntegerQueue;
};
```

Access Modifiers in C++

```
class Node{

   private:
      int element;
      Node *next;

      // the constructor is private!
      Node(int theElement, Node*n)
         : element(theElement), next(n){ }


      // only the enqueue function from the
      // IntegerQueue class is a friend
      friend void IntegerQueue::enqueue(int x);
};
```

Life Before Exceptions

- C++ was designed to be efficient and fast
- Exception handling before "exceptions"
  - `abort` – immediately terminates the program (not graceful)
  - `exit –` slightly better, calls destructor of static objects, pass an int
  - `errno` – a bit that can be set to indicate various errors
- Assertions
  - Use `assert` keyword defined in `cassert` library
  - Use `#define NDEBUG` to turn off assertions

Exceptions with File I/O

- Read this article:

http://gehrcke.de/2011/06/reading-files-in-c-using-ifstream-dealing-correctly-with-badbit-failbit-eofbit-and-perror/