LECTURE 30: CLASSES IN C++

Today

- Reading
  - Weiss Ch. 4 (4.1-4.4)
- Objectives
  - More memory management in C++
  - Classes in C++

Announcements

- Quiz this Friday on C++
- Final is on Friday May 15th at 9am

This week's assignment

- Implement a priority queue in C++
- The header file is provided as starter code
- Use Aquamacs and Terminal
- Today's lecture and lab will help prepare you
- "Java Structures" provides you already with an implementation for a priority queue (see the book)
- C++ Library Reference (link on course webpage)

Recap

- *Similarities between Java and C++?*
- *Differences?*

Memory Management

- In Java, most types are objects
  - Everything except for primitives allocated using `new`
  - Memory is taken from the heap
- In C++, everything is a primitive
  - Allocated on the stack not the heap
  - Allocate from heap by explicitly using the `new`

Memory Management

- *Changes the semantics of assignment*
  - Assignment (=) means copying
  - But now you have to think about *what* you're copying
  - Assignments happen more than you realize!

Classes in C++

- See `IntCell` class in `intcell_onefile.cpp`
- Classes in C++
  - Class declaration ends with semicolon!
  - Visibility (public, private) declared for sections
  - Default is private
- Can only use `IntCell` class in same file!

Classes in C++

- Move class definition to header file (.h) so it can be imported
- Even better: separate the declaration of the class from the implementation
- Only need to include the header file (.h) to compile a user's code
- Can change implementation without recompiling user's code

Classes in C++

- Include files can include more include files…
- Code won't compile if include a file more than once!

```
#ifndef INTCELL_BEST_H
#define INTCELL_BEST_H
// class declaration
#endif
```

- If variable `INTCELL_BEST_H` is already defined then won't include the class declaration
- Always do this when defining class declaration in header file!

The const keyword

- Accessor method vs. mutator method
- Using `const` tells the compiler that the function is an accessor
  - Promise function will not change the state of the object
  - Allows compiler to optimize your code

Destructors

- No garbage collection in C++
- Any memory you allocate from the heap, you must free!
  - Otherwise, you have a "memory leak"
- The destructor is called when the variable goes out of scope
- Name of the destructor is `~ClassName`

Assertions

- To write an assertion,

```
#include<cassert>  // need to include this
...
assert (boolean_expression);
```

- Can turn off assertions if you write

```
#define NDEBUG
```

  before the `#include<cassert>` statement