Computer Science 62
Lab 11

Wednesday, April 16, 2014

In this lab, we will be playing with pointers by building a linked list class similar to our Java implementation. In addition, we'll also look at a few experiments to understand how memory needs to be managed in Java.

Getting started

Copy all of the contents from:

```
/common/cs/cs062/labs/lab11
```

On the course web page, look at our notes for lists and refresh your memory on how we implemented linked lists in Java.

Take a look at the definition of the `node` class in C++ (both the .h and .cpp) files. What is different?

Compile the `node` class by typing:

```
g++ -c node.cpp
```

(notice that we're just compiling the file, but not building an executable yet).

LinkedList

Now, take a look at the linked list header file (`linkedlist.h`). The header file contains a basic set of linked list methods. Implement these methods in a file called `linkedlist.cpp`. Most of these methods should be a straightforward translation of the code from Java. Note, however, that you will need to use pointers! Just to keep you in the good habit, `#include <cassert>` and use assert statements appropriately in your code.

Again, to compile this type:

```
g++ -c linkedlist.cpp
```

Using the linked list class

Once you have it compiling and you think you have it working, look at the `linkedlisttest.cpp` class, then compile it:

```
g++ -c linkedlisttest.cpp
```

and then compile/link all of your previously compiled object files into an executable binary (notice that `linkedlisttest.cpp` has a `main` method, which is required to construct an executable):

```
g++ -o linkedtest node.o linkedlist.o linkedlisttest.o
```

or

```
g++ -o linkedtest *.o
```

Run the test:

```
./linkedtest
```

If all works well, you should the numbers from 0 to 9 printed out, except 4 is exchanged with 100.

Note that anytime you change a .cpp file, you'll need to recompile that particular file, but then also recompile the executable with the -o command. Eventually, we'll talk about makefiles which makes this process a bit faster.

Things that make you go "hmm..."

Change the `main` method in `linkedlisttest.cpp` to run test2 and recompile. Repeat for test3 (it's a little weird to store a `Node` in a `vector` since we're ignoring the `nextElement` pointer, but I wanted to convince you that test2 and test3 are roughly the same). Do these results surprise you? Why are these results different?

Try a few different varieties of test2:

- Change "`LinkedList l2 = l`" to "`LinkedList l2(l)`". Does this change your result? Where does that constructor come from?
- Change the linked list variables in test2 to be linked list pointers. Use the "`new`" operator to create a new linked list object for l. Set l2 = l. Does this change your answer? Does this make sense?