Data Structures with C++ : CS189

Lecture 5-1: Lists

Recap

- A vector is an array that manages its own memory
- Data gets push_back'd to add to the array
- If the array is full, Reserve is where the vector re-allocates more memory for itself
- As a class, it can also error check any reads or writes to prevent crashing
  - Crashing is worse than wrong

List

- Now instead of an array, picture a chain
- Each link in the chain holds one T
- You hold on to the two ends of the chain and not the middle links
- What can this ADT do?
    - Find first or last
    - Add to front or back
    - Remove front or back
    - Throw the chain away
    - If you know where a link is, you can remove that link

List's Contract

http://www.cplusplus.com/reference/vector/vector/

http://www.cplusplus.com/reference/list/list/

- The word List has no meaning since it is abstract. So we've decided that something with these methods at these speeds are "List"
  - Size in O(1)
  - Push or Pop Back in O(1)
  - Push or Pop Front in O(1)
  - get Front or Back in O(1)
- Sounds better than Vector?
  - get data At location in O(n)
  - Find data in sorted list O(n)
  - Vector does those two in O(1), but does PushFront, Insert, and Erase in O(n) instead of O(1). Find in O(log n)

Implementation vs Contract

- The definitions of List and Vector have no code in them
  - Just results and Big O
- The details of how you write them are up to you
  - Information Hiding!
- STL is the fastest speed with smallest memory because it's been worked on for decades
- So we are going to look at *one* possible implementation of List

Nodes

```
template<typename T>
class List
{
        struct ListNode
        {
        ListNode *mPrev;
        ListNode *mNext;
        T mData;
        };

        ListNode *mHead;
        ListNode *mTail;
};
```

- Node is another abstract word with no real meaning
- Continuing with the chain analogy, a node is a link in the chain
- This struct is an "inner class" since it is defined inside the List class itself
  - That would normally make the name of the class "OuterClass::InnerClass" if in a main
- But the struct is also private. It's just for us
  - Information Hiding!

Go to Canvas

College Tip: You should at least skim the chapter of the textbook a class will cover *before* that class. The textbook is really useful, but admittedly hard to read. It's worth it to be able to ask questions on Tuesday.

Sentinel Nodes

```
List()
{
mHead = new ListNode;
mTail = new ListNode;

mHead->mNext = mTail;
mTail->mPrev = mHead;
}
```

- When you have pointers, you have potential crashes
  - Pointer to deleted node
  - Node that links to nothing
- With no data in the List, Head and Tail would be null
  - Every operation would have to check for null
- A Sentinel Node is like a "hook" for the chain. It's a node that will always exist, but has no data
- Now every real node knows that there is definitely a node on each side.

Pointer Management

Reminder:
ALWAYS DRAW A PICTURE!

- Even with Sentinels helping with nulls, you can still make problems yourself
- There is one rule that will help you through this entire semester
  - DRAW A PICTURE
- Draw a picture with boxes for nodes and arrows for pointers
- To add a new box and connect it in to the list you need to draw 4 arrows
  - So your code better have at least 4 lines
- Removing a box requires 2 arrows

End

List is the first of many that use Nodes and lots of dynamic memory