In-class objectives

Learn more about templating and data hiding by implementing a simple container adapter and solving a programming problem using it.

Part 1

Start developing a templated container adapter (based on an STL container) for a circular, doubly linked list. To start, discuss and briefly implement the following using either Stack/Queue or CardDeck as a starting point:

- void insertFromFront (T &); // inserts an element of type T in the front
- void insertFromBack (T &); // inserts an element of type T at the back
- void deleteFront ();    // removes the first element in the list
- void deleteBack ();    // removes the last element in the list
- bool isEmpty();       // returns true if and only if the list is empty
- ~cdl_list ();    // is this really needed?

Part II: Abstracting an “iterator” for your new container

To be doubly, you should be able to move forward and backward. To be circular, the first element should consider the last element its preceding neighbor and the last element should point to the first element as its “next” neighbor.

Discuss adding the ability to move around your templated cdl_list by adding the following “iterator”-like functionality. You should create an internal, private data member called “current” that indicates the node currently visited by the iterator.

What are the special cases you need to consider and why?

HINT: This exercise is much easier if you used a container that supports random access. Discuss why and/or how? Since we are encouraging data hiding in this exercise, any viable approach will receive full credit.

- void reset();    // makes the “current” pointer the first/front element
- void next();     // makes the “current” pointer point to the next node neighbor
- T currentValue(); // return the data in the node that the “current” refers to
- void delete();   // delete the node currently visited; current = old->next
- operator++()     // how is this different than next()?
Part III: Putting it all together

In ancient times, a fair princess Buttercup had many suitors. Thinking her true love dead, she decides to use the following procedure to choose the knight she will marry. First, all suitors are lined up and given a number from 1 … n. Then, because of a strange fascination with the number three, she would count to three and remove the third suitor she considers that round from contention. This process would repeat, returning to the front of the list once all suitors were considered. For example here is the results of the process with n = 6:

1 2 3 4 5 6 (3, you’re not the one for me)
1 2 4 5 6 (6 … sorry)
1 2 4 5 (4, no more)
1 2 5 (2, I don’t choose you)
1 5 (5, at least you’re still alive)
1

Luckily, Buttercup’s true love Wesley was not killed but rather became a protégé of the Dread Coder SJE. As part of the Dread Coder’s crew, you have been asked to help Wesley determine which position in line he should stand to win Buttercup’s hand.

1.) How could you use the circular doubly linked list, and specifically Part II, to perform the simulation and display what spot Wesley should stand in for any arbitrary n number of suitors. Add pseudocode if you have time

2.) Write brief text on the back on how you used data hiding for this exercise. Are there any special advantages?