CS 240

Data Structures and Algorithms

Spring 2014

1 Lab 07

1.1 Goal

The goal of this assignment is to learn to use and understand templates as well as familiarizing yourself with the difference between a Singly Linked List and a Doubly Linked List.

2 List ADT

Recall that we have discussed a List ADT, and that a Linked List should implement this ADT. Following is the List ADT as defined in the textbook.

Collection of Data Elements:
A sequence with a finite number of data items, all of the same type.
Basic Operations:

- Construction: Create an empty list
- Empty: Check if the list is empty
- Insert: Insert an item into the list
- Delete: Remove an item from the list
- Traverse: Go through the list or part of it, accessing and processing the elements in order. This operation is also referred to as iterating through the list

Note: (You are not required to implement all of the traversal behaviors.) Traversal encompasses some general behaviors such as:

- Search a list: Traverse the list, examining elements (or some key field in them) for a particular item, stopping when it has been found or the entire list has been checked
- Output a list: Traverse the entire list, displaying each list element
- Copy a list: Traverse the entire list, copying the list elements into another list
- Save a list: Traverse the entire list, writing each list element to a file

3 Implementation

For your implementation you will be required to implement slightly more than the basic operations of the List ADT. You will be required to implement the following functions with the following names.

- Constructor :

  `DLinkedList`

  - Creates an empty doubly linked list
- Destructor :

  `~DLinkedList`

  - Deletes a doubly linked list
- Empty :

  `empty`

  - Determines whether or not a doubly linked list is empty
- Insert (at first or at last, by value) :

  `insert_first`

  - Inserts some element (that is passed by value) into the first position of the list

`insert_last`

  - Inserts some element (that is passed by value) into the last position of the list

- Delete (from first or from last or by value) :

`delete_first`

  - Deletes an element from the first position in the list, and returns the deleted value

`delete_last`

  - Deletes an element from the last position in the list, and returns the deleted value

`delete_value`

  - Deletes any element from the list that contains the passed value, and returns the count of elements that have been deleted.

- Count :

`count`

  - Counts how many data elements are contained in the linked list

3.1 Notes

You are required to implement the extended Linked List ADT as defined above, using the method names indicated. Your method signatures should make sense with respect to what is necessary to pass to the methods as well as their return types. Additional classes and methods should be implemented as required.

In order to make this Linked List class more generic, it will be required to use template programming.

For your class(es) in this assignment you will likely only need one templated parameter, as such your class declaration should look like this

```
template <typename Element>
class DLinkList
{
     // ...  members of DLinkList ...
};
```

Note that the word Element is a generic type parameter naming a data type to be stored in the container class `DLinkList`. The keyword `typename` may be replaced with `class`. The keyword `template` specifies that what follows is a pattern for a class, not an actual class declaration. Unlike function members of regular classes, definitions of function members of a class template must be available to the compiler wherever the class template is used.

There are two ways to accomplish this, but for the purposes of this assignment, your templated code implementation is to be kept in a `.cc` file instead of a `.cpp` file, and your header file must have an include statement that connects it to the appropriate `.cc` file.

Note: You may use your code from last week's lab as a jumping-off point. It should be straightforward to make the modifications necessary in order to accomplish a templated doubly linked list. You can, if you so choose, re-write everything from Lab 05 and Lab 06 and start fresh. There are no class requirements other than the ones setforth in this document regarding `DLinkList`.