TO COMPILE: type: g++ linklist.cpp Driver.cpp

Program 6

Approach: Following the suggestion on the OneNote, the first 3 methods that I think of are insert to end, insert and remove. For the remove method, I decided to split it into 2 others method that can remove at an index and remove an integer from the list.

For the copy constructor, I realize that I will need to add an operator overload method. And for the destructor, I will also need a clear method as well.

I decided to check the STL so see if there are any other methods that I should add. And I realized that I will also need to add a method to check if the list is empty. And another method to see how many elements are in the list

Last, I added a method to print out the content and 2 reverse method.

There are multiple ways to implement the Node structure. I have decided to use a head Node directly instead of a dummy head Node or a Node at the end. To compensate for this, I will need to check for head node if it is NULL at almost every method.

Insert an integer to the list is fairly simple. I decided to use an index approach to do this. For this assignment, I use the structure 1,2,3,4 instead of regular index like 0,1,2,3. And when I insert, I stop at a location before the designated index. So, if I want to insert at index 2, I stop at 1. Then I fix the pointer from 1 to the new Value. New Value to current index 2. And set new value as index 2.

For the copy constructor, I remembered from the lecture that we need to avoid for self-assignment. So, I check for that and clear the old list. Iterative through the list that need to be copy and create new value.

For the remove a specific integer, I decided to iterate through the entire list and remove all integer that has that value instead of removing just the first occurrence.

Since this list only hold integer, I see no benefit to add a method that will only find the integer and not doing anything. If we want to get an integer, just pass it into the remove method and remove method will tell if it successfully finds and remove or not.

Iterative reverse approach.

For this approach, I decided to use 3 pointers to do this. I will iterate through the list, fix the current Node to point to the previous. Update to go through the next Node and set the previous as the current one so that I can rearrange in next loop.

Recursive approach

For this approach, I decided to recursively go to the end of the list. From that point, I change the head reference to the end. And continuously rearranging the current pointer. I used a double pointer to rearrange the element directly instead of having to return a Node and change it later.

So, the difference is: for iterative, I start at the front and go through the entire list, rearranging pointer along the way. For recursive, the work only started when we reach the end element, at that point, the stack for the recursive function will pop and we will reassign the pointer.

Reference

<http://stackoverflow.com/questions/7271647/what-is-the-reason-for-using-a-double-pointer-when-adding-a-node-in-a-linked-lis>

// The one above explain to me why my recursive function won’t update itself

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

// I check the one above for ideas about what methods should I choose

<http://stackoverflow.com/questions/5856628/program-received-signal-sigsegv-segmentation-fault>

// I check that one when I receive an error on Linux, even though on Window it still compiled

Textbook: Data Abstraction and Problem Solving with C++, Walls and Mirrors

// I also check the textbook for what methods I should choose