Hugh A. Miles II

09/21/16

COP3530

Circular Linked List

* Summary
  + My Circular Linked List a basic Linked List Class, but instead of the last Node (tail) pointing to nothing its points to the head. This is a templated so the Nodes can hold any type based upon the type templated at its instantiation. Each node has its own index, a bool value indicating whether it’s the head or not, and the address of the next Node in the list.
* Structure
  + Node<type>
    - Constructor (type initValue, int strIndex)
      * In the constructor you need to indicate the value to be set, and where the Node is being placed with the strIndex.
    - Int index
      * Location in the list
    - Type value
      * The data value stored in side the node.
    - Node<type>\* nextNode
      * Address for the next Node in the list
    - Bool isHead
      * Value that indicated whether you are at the front of the list.
  + LinkedList
    - Head<type>
      * Pointer to start of the list (index = 0)
    - Int curLength
      * Keeps track of the number of Nodes in the current list.
* Methods
  + Insert (type value, int index)
    - Allow users to insert element at a certain spot in the current list that is less than the this.length + 1.
  + DeleteNode(int index)
    - Allows user to delete Node at any point in the current list that is less than this.length + 1
  + Print
    - Prints entire list sequentially.
  + PrintAtIndex(int index)
    - Prints element at index given in the parameter
  + Helper Methods
    - Append(type value)
    - ResetIndexs
      * Since each node contains its own index. This method resets the node.index for each node in the list after a delete or insert.
* Test Cases

1. Small Set (4 elements)
   1. input\_small.txt
   2. Output
2. Medium Set (50 elements)
   1. input\_medium.txt
   2. Output
3. Large Set (100 elements)
   1. input\_large.txt
   2. Output