WAYNE STATE UNIVERSITY

COLLEGE OF ENGINEERING

Computer Science Department

CSC 2201: Computer Science II – Lab
Lab5

Description:

You will implement a list ADT using a singly linked list.

Goals:

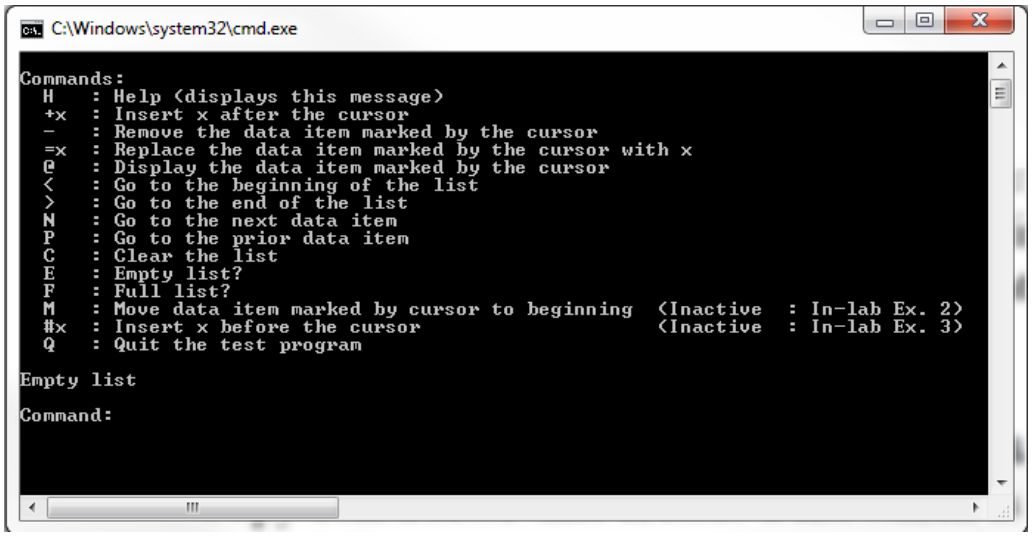
Learn how to implement linked data structures using C++ pointers.

Where to Start:

1. Download Lab5_Work.zip from Blackboard
2. Unzip the file
3. Open the solution in Microsoft Visual Studio
4. Make sure that the project does not show any compile errors.
5. Implement methods and operations of the ListLinked class:

   5.1 Implement `List<DataType>::ListNode::ListNode(const DataType& nodeData, ListNode* nextPtr)`

   The default constructor that creates a list node containing item nodeData and next pointer nextPtr.



   5.2 Implement `List<DataType>::List(int ignored)`

Another constructor that creates an empty list. The argument is included for compatibility with the array implementation (maxSize specifier) and is ignored.

5.3 Implement `List<DataType>::List(const List& other)`
Copy constructor that creates a list which is equivalent in content to the "other" list.

5.4 Implement `List<DataType>& List<DataType>::operator=(const List& other)`
Overloads assignment operator. Sets the list to be equivalent in content to the "other" list.

5.5 Implement `List<DataType>::~List()`
Destructor. Deallocates the memory used to store the nodes in the list.

5.6 Implement `void List<DataType>::insert(const DataType& newDataItem) throw (logic_error)`
Insert newDataItem to the list after the cursor. If the list is empty insert newDataItem as the first data item in the list. Always moves the cursor to newDataItem (new inserted item).

```
Empty list

Command: +x
Insert x
[x]
```

5.7 Implement `void List<DataType>::remove() throw (logic_error)`
Removes the item marked by the cursor from a list. Moves the cursor to the next item in the list.

```
Command: +g
Insert g
x y d [g]

Command: -
Remove the data item marked by the cursor
[x] y d
```

5.8 Implement `void List<DataType>::replace(const DataType& newDataItem) throw (logic_error)`
Replaces the item marked by the cursor with newDataItem and leaves the cursor at newDataItem.

```
[x] y d

Command: =j
Replace the data item marked by the cursor with j
[j] y d
```

5.9 Implement `void List<DataType>::clear()`

Removes all the items from a list. Sets head and cursor to 0.

```
Command: c
Clear the list
Empty list
```

5.10 Implement `bool List<DataType>::isEmpty() const`

Returns true if a list is empty. Otherwise, returns false.

```
Command: e
List is empty
Empty list
```

```
Command: +X
Insert X
[X]

Command: E
List is NOT empty
[X]
```

5.11 Implement `bool List<DataType>::isFull() const`

Always returns false. It is a linked list and it can never be full.

```
Command: F
List is NOT full
[X]
```

5.12 Implement `void List<DataType>::gotoBeginning() throw (logic_error)`

If a list is not empty, then moves the cursor to the beginning of the list. If list is empty, throws logic error.

```
Command: <
Go to the beginning of the list
[X] K F G H J
```

5.13 Implement `void List<DataType>::gotoEnd() throw (logic_error)`

If a list is not empty, then moves the cursor to the end of the list. If list is empty, throws logic error.

```
Command: >
Go to the end of the list
X K F G H [J]
```

5.14 Implement `bool List<DataType>::gotoNext() throw (logic_error)`

If the cursor is not at the end of a list, then moves the cursor to the next item in the list and returns true. Otherwise, leaves cursor unmoved and returns false.

```
Command: <
Go to the beginning of the list
[X] K F G H J

Command: N
Go to the next data item
X [K] F G H J
```

5.15 Implement `bool List<DataType>::gotoPrior() throw (logic_error)`

If the cursor is not at the beginning of a list, then moves the cursor to the preceeding item in the list and returns true. Otherwise, returns false.

```
Command: N
Go to the next data item
X [K] F G H J

Command: P
Go to the prior data item
[X] K F G H J
```

5.16 Implement `DataType List<DataType>::getCursor() const throw (logic_error)`

Returns the item marked by the cursor. Requires that list is not empty.

```
Command: @
Element marked by the cursor is G
X K F [G] H J
```

5.17 Implement `void List<DataType>::showStructure() const`

Outputs the items in a list. If the list is empty, outputs "Empty list". This operation is intended for testing and debugging purposes only.

6. Activate all tests in the program test5.cpp by changing the definition of `LAB5_TEST2` and `LAB5_TEST3` from 0 to 1 in config.h and recompiling.
7. Implement the following methods

7.1 Implement `void List<DataType>:: moveToBeginning() throw ( logic_error )`

Removes the item marked by the cursor from a list and reinserts it at the beginning of the list. Moves the cursor to the beginning of the list.

```
Command: P
Go to the prior data item
X K F [G] H J

Command: m
Move the data item marked by the cursor to the beginning of the list
[G] X K F H J
```

7.2 Implement `void List<DataType>:: insertBefore ( const DataType &newDataItem ) throw ( logic_error )`

Inserts newDataItem before the cursor. If the list is empty, then newDataItem is inserted as the first (and only) item in the list. In either case, moves the cursor to newDataItem.

```
Command: N
Go to the next data item
G X [K] F H J

Command: #L
Insert L before the cursor
G X [L] K F H J
```

8. Test your implementation using the program in the file test2.cpp.

Create a Zip file of your solution:

1. Right click on your solution in Solution Explorer
2. Click on “Open Folder in File Explorer”
3. Go one level up in file explorer
4. Right click on your solution folder
5. Add it to archive by creating a zip file

Upload the zipped file on Blackboard:

1. Go to Blackboard
2. Click on this course (CSC 2201: Computer Science II Lab)
3. Go to the folder “Labs”
4. Click on the “Lab1_Work” assignment
5. Upload your zipped file