HW02

Omar Elmady

Purpose

This program implements a class CharLinkedList with properties similar to std::vector, including modification, concatenation, and printing.

Acknowledgements

I got help from some of the staff at office hours.

File descriptions

The files uploaded contain all the code used to compile and run the program.
CharLinkedList.h: contains the header file defining the functions and properties of the class, including private helper functions used for implementation.
CharLinkedList.cpp: contains the implementation of the functions defined in the header file.
Unit_tests.cpp: contains the driver for the file as well as some simple print statements to clarify use of the class.
Makefile: contains the instructions for compiling the program using the terminal.

Compilation

Simply write to terminal **make test_list**, possibly preceded by **make clean**.

Data Structures

The comparative advantage of linked lists over array lists is their efficiency in adding or removing elements; namely, they do not have to allocate or copy data beyond the amount necessary to store the elements. Their disadvantage is in traversal. For example, adding a character to the end of a list would require traversing the entire list, taking linear time as the list grows linearly… whereas arrays can access indices in constant time. Positioning pointers at the front and back of the linked list, where most of the functions modified, made the code easier to write and made the program more efficient.

Testing

Unit testing was very valuable in helping me slowly but surely diagnose and target an error. However, it is often the case that flaws in earlier functions can lead to errors in later functions that call them. I came across a few problems regarding my linking of the lists, which I had to address several times. If I could have ensured my code was correct, I would have saved time debugging. I look forward to learning more strategies to ensure the correctness of code. In the meantime I was happy to practice debugging.

Q1: What was easier about linked lists than array lists?

Honestly this assignment was more conceptually difficult as well as more involved than the previous. My code ended up being longer as a result, and I wasn't always able to get elegant solutions. Managing pointers, allocating nodes, and modifying data all was more difficult. Using recursion to simplify some of the function calls made my life easier. I would also say that debugging was easier and less stressful because I have more practice.

Q2: What was harder about linked lists than array lists?

I would say that managing pointers and making sure that modifications did not break the links of the list was the hardest part. It's like constructing a chain-link fence vs a wooden fence.

Q3: How might a client implement a LinkedList from ArrayList code?

To translate the ArrayList code to a LinkedList, they would have to create and modify contiguous heap memory instead of node instances. Beyond that, the two programs are quite different because the LinkedList depends on pointers whereas the ArrayList uses indices.