Nicholas Ang

CS 202 - 1101

Project 8 Documentation

Purpose: The project gets us to create a test driver and implement classes from the headers given to us. The whole project is based on nodes and dynamic allocation with arrays. The purpose of this project is to create and utilize dynamic memory and properly allocate and deallocate it. Additionally, it uses aspects of previous knowledge such as pointers, string manipulation, iostream, along with other key aspects of C++. It touches on pointers and pointer manipulation to manipulate the node lists and array lists.

Design: The project is split up into two main folders with the source code and the libraries/header files. The proj8.cpp file is the test driver that I created for the project. It uses all the functions from the classes that are implemented from the secondary files. The ArrayList class implementation has many methods. It has 3 constructors, 1 destructor, and 12 other functions to supplement the class. The constructors construct based on the values given and work as a default, parameterized, and copy constructors. The destructor clears everything out and deletes the dynamically allocated space, freeing it. The front and back functions find the first and last parts of the valid array. The size function returns the valid size, empty function shows whether or not the array is empty, and the clear function clears out the array. The assignment operator = works similarly to a copy constructor and assigns the value of one array to another. The find function finds the targeted array and returns it. The insert after and insert before functions add an entry to the array while the erase function deletes an entry from the array. Two operators are overloaded which are the [] and the << operators. The [] operator is overloaded to get the position of an array entry and return it. The << operator is overloaded to output all the entries in the array. Additionally, there is the resize function which allows the array to be reallocated to be bigger or smaller. The Node class is implemented with similar functions and methods. It also has 3 constructors, 1 destructor and 12 other functions. The constructors construct based on the values given and work as a default, parameterized, and copy constructors. The destructor clears everything out and deletes the dynamically allocated space, freeing it. The size function returns the valid size, empty function shows whether or not a node is empty, and the clear function clears out the node. The insert after and insert before functions add a node to the node chain while the erase function deletes a node from the node list. The find function finds the targeted node and returns it. Throughout the program, there are many precautions in place to prevent the program from breaking such as checking for NULL and resetting values to NULL/nullptr. Two operators are overloaded which are the [] and the << operators. The [] operator is overloaded to get the position of a node and return it. There is also another version of the [] operator for const values. The << operator is overloaded to output all the entries in the node. The front and back methods return the head of the node chain and the back of the node chain. The assignment operator = works similarly to a copy constructor and assigns the value of one node to another. The test driver uses all of these methods to test out node chains and array lists.

Problems/Challenges: This project was one of the most challenging ones. There were segmentation faults everywhere and the program would not work as intended. Specifically there were many problems with the insert after and insert before where it would not return the proper pointers or would break the console. I also had a problem with the find function that entirely broke the program but I realised that I forgot to increment i. I felt this was one of the most time consuming and difficult challenges as nodes and node manipulation was recently covered.

Possible Changes: I would try many changes in this project. Although everything works, some of it is not that clean looking and proper. I would like to experiment more with nodes and node chains/links because that is pretty interesting and I really had to think about where things would go and where I can move the node. I would also try to experiment with more dynamic memory because dynamic memory has a lot of usages and little things about it that are hard to understand.