Data Structures Lab 8

Joseph Schnizer

Gabe Imes

Eric Bridgens

In this lab, areas explored included linked lists, object-oriented programming, and pointers. Linked lists were explored in depth as a use for the organization of data, in this case different parts. These parts were represented as objects that hold different descriptive information. Linked lists are extremely important and can be applicable to a professional setting due to their focus on the core of data organization and efficiency in accessing and working with different elements. Object oriented programming also proves to be important in this application because it allows you to store a set of information in each node instead of just one piece of information per node. This applies to the professional setting as it allows for more efficiency and readability in design. Finally, pointers are focused on in conjunction with linked lists as they are what make them possible. Without understanding the fundamentals of pointers, linked lists are difficult to implement. In a professional setting, pointers are important because they allow for dynamic memory allocation which can allow the programmer to do some useful things at runtime. Most importantly, pointers make the functionality of linked lists possible.

For task 4, when creating our display function for our part, we decided as a group that the members we should include were the SKU number, the price, the description of the part and the quantity. We felt that these were the most appropriate members to display to the user. The SKU is the code that details what the part number is which is the main detailing member for a part. We chose to print out the description of the part because explaining to the user what that part is informs the user helps the user decide if they need that part. The price is also important to present to the user because letting our users know the price of a part is critical to informing the user on whether they want the part or not based on the cost. Lastly, the quantity was used because letting the user know how many of the parts are in store helps the user determine how many parts can be purchased.

In terms of rewriting code to make task 4 successful, we did not have to rewrite and change any code from previous tasks. The only changes we made to both classes were the display functions themselves, then including “Part.h” into the DoubleLinked.h file to access the parts Display method. The code that we could reuse in Task 4 would be the display function for the DoubleLinked class as the member could work for any other class as long as it’s properly declared in the file.

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generated