Project 2 Reflection

Hailey Wilder

Design Description

In project 2, a grocery list was maintained. Items could be added, deleted, and summed. In order to achieve this task, I utilized an item class and a list class which were referenced in a main and printed via switch function from the main into the command window.

The item class stores item names, units of the items, quantity, single item price, item total, and purchase total. In the implementation of item, there is the option to pass each of the afore mentioned variables from user input. The load constructor gives the option to create items and the print option prints item info to console.

The list class stores an item count, memory (increased as needed in increments of 4), and address pointers. In the implementation of list, the addItem adds items to the dynamic array and deleteItem deletes items. There is a bool check so if an item already exists in the array, it cannot be added multiple times. The options getCount and getTotalCost return total items and cost, respectively. Finally, printList prints the list to the console.

Test Plan and Results

Input	**Expected Outcome**	**Observed Outcome**
Std::cin.ignore(256, '\n'); std::getline(std::cin, name);	Store user input into name	This method allowed user input with spaces

If(itemList == name)
Else
itemList.addItem(item)
	this uses the overloaded "==" operator to compare to see if the item name already exist in the list	will not add if item name is in the list, else item is added to itemList

while (item > itemList.getCount() || item < 0 || std::cin.fail())	checks user input to make sure they entered an item that is in the list.	Deletes the item and later reduces item index by one
List::addItem(Item item)	Adds item to the dynamic array	Stores items up to memSize (which starts at an array of 4 and increases by 4). Transfers items between arrays. Releases memory allocation.
List::deleteItem(int)	Deletes item from list	Removes a specific item from the list shopList. Decreases the counter. Releases memory allocation.
List::getCount()	Returns number of items in the list	Returns the counter variable (dynamic array size) when called.
List::getTotalCost()	Adds prices for each item and displays total	Adds the price for each item and returns the total when called.
List::printList()	Prints the entire list	Prints the list and the total sum of the list.
Item::getName()	Pass item name	Returns item name
Item::getUnit()	Pass item unit	Returns units of item

Project 2 Reflection

Hailey Wilder

Item::getNumToBuy()	Pass item quantity	Returns quantity of item
Item::getUnitPrice()	Pass item price	Returns unit price of item
Item::getTotal()	Pass total	Returns total price
Item::print()	Print item information	Prints item information to console

Reflections

This project seemed to come to me much more intuitively than the previous project. The previous project had a component of continuous updates which I found difficult while this project was more array/list based. In order to solve this problem, I just needed to get user input and solve equations. This seemed simpler to me, but I think that is an inherent part of programming – trying to find ways to understand the task at hand. I was more successful at that this project than I was last project.