Christopher Merrill
CS162
10/23/2016

Assignment 2 Reflection Document
(Design, Test Plan, Test Results and Reflection)

ORIGINAL DESIGN PLAN

Requirements:
Create a grocery list
Add items
Remove items
Display the list
Display the total cost of all of the items
Allocate more memory as necessary

Display:
Format the grocery list entries in the following way:
“Quantity” “Units” “ Item Name” ($”Unit Price” per “Units”) Total: $”Extended Price”

Example:

****
GROCERY LIST
****

3 box Apples ($5.00 per box) Total: $15.00
1 gallon Milk ($2.00 per gallon) Total: $2.00
2 liter Coca Cola Soft Drink Beverage ($1.00 per liter) Total: $2.00

GRAND TOTAL: $19.00
****

What would you like to do?

1. Add Item
2. Remove Item
3. Exit

Item Class:

Overview:

Used to hold the information for items within the list. Contains info for item names, quantities, units, price, and extended price. Get functions will be needed for extended price to calculate the grand total, and name to compare the names of entered items.

Data elements:

- string name
- string unit (can, box, pound, ounce, etc.)
- int quantity
- double unitPrice
- double extendedCost

Functions:

- double getExtendedCost()
- string getName()

List Class:

Overview. The list class will manage the actual grocery list, including adding items, removing items, and allocating more memory when necessary. It will contain the list itself, the size of the list and the grand total of all items on the list as data elements.

Data elements:

- Item** list
- int size
- double grandTotal

Functions:

- void addItem()
- void removeItem()
- void allocate
- double getGrandTotal() : returns the grand total

Main.cpp

Create a list object

DO the following while the user's choice is not to exit the program

- Allocate more memory if necessary
- Display the menu
- If the user wants to add an item
  - Ask the user to enter the item name
    - Check to see if that item is already in the list (with overloaded == operator)

Ask the user to enter the desired quantity (validate input is an integer and >0).
Ask the user to enter the units
Ask the user to enter the unit cost
Add item to the list (addItem function)
Calculate the new grand total (getGrandTotal function)
Clear the console
Print the grocery list header
Print the list
Print the grocery list footer which includes grand total
Add one to the size of the list

If the user wants to remove an item
Ask the user to enter the name of the item they want to remove
Check to see if that item exists
Remove the item (removeItem function)
Calculate the new grand total (getGrandTotal function)
Clear the console
Print the grocery list header
Print the list
Print the grocery list footer which includes grand total
Subtract one to the size of the list (decrementSize function)

If the user wants to exit the program
Exit the loop

TEST PLAN and TEST RESULTS:

Test	Results	Comments
Spaces in names: Add an object to the list that contains the following name and units, which contain spaces: Name: Whole Wheat Bread Units: Plastic Bag	Correctly added the objects to the list and displayed them on the console. No issues with cin afterwards when adding new items.	No changes necessary. (Using getline along with cin.ignore and cin.clear seemed to do the job).
Extended Prices: Add items with the following quantities and unit prices:	Item 1: Extended price was $15.00 Item 2: Extended	No changes necessary.

Item 1: 3, $5.00 Item 2: 2, $10.00 Item 3: 5, $20.00 And verify the extended prices are correct.	price was $20.00 Item 3: Extended price was $100.00	
Grand Total: For the items added above in the extended prices test, verify that the grand total is $135.00. Then remove item 2 and verify that the grand total is $115.00	Grand Total Before Removal: $135 Grand Total After Removal: $115	No changes necessary.
Duplicate Entry: Enter an item with name “a”, then attempt to enter another item with name “a”.	Unable to add second item with duplicate name, exception was thrown and menu was presented again.	No changes necessary, this was the intended result.
Removal of Non Existing Item: Try to remove an item with a name that doesn’t exist in the list.	Informed that “Item does not exist” and the main menu reappeared	No changes necessary, this was the intended result.
Memory Allocation: Attempt to add more than 4 items to the list and see if it is allowed (which would mean that additional memory was allocated)	Able to add upwards of 50 items with no issues. Program only adds 4 extra slots at a time, so allocation is working as expected.	No changes necessary.

REFLECTION:

Overall my original design held up, meaning I didn't have to make too many changes in order to get my program to work. I did however have to add a few additional member functions to my List and Item classes. I had to add get functions for the rest of the Item class data members, which I used when duplicating arrays for memory allocation or item removal. I also added a print function to Item to make the list display a little easier. In the List class, I added a "print list" function which helped display the grocery list, as well as a member function used to overload the == operator. As far as main.cpp, I didn't have to make any significant changes to the design flow. Although I did move some of the processes from main.cpp to the specific member functions when it seemed like a better fit.