CS 162 – Assignment 2
Grocery List

Daniel Beyer
Dan.beyer@gmail.com
04/19/2016

Requirements

Design, implement, and test a grocery shopping list program. The program should maintain and display a list of items. The program will have 2 classes: Item and List. Item should have data elements for: item name, units, number to buy, and unit price. List will use a dynamic array to hold the item objects, starting with a capacity of 4 items.
The program must be able to create a list, add items, remove items, and display the shopping list. The user will be prompted to enter each data element of the item, and the display of the list should show each element in addition to the extended total price of the individual items and the total price of the entire list. The program will also use an overloaded "==" operator to test if an item is already been added to the list.

Program Input

User enters:

- Initial Menu:
    - 1. Add item to grocery list
        - User Enters:
            - Item name (string)
            - Item unit (string)
            - Unit quantity (int)
            - Unit price (double)
    - 2. Remove item from grocery list
        - User Enters:
            - Int selection from grocery list
    - 3. Print grocery list
    - 4. Exit

Program Output

Program displays the Grocery List. For each item in the list, the following information elements are displayed: Item name, item unit, quantity, price, and total cost for the item.
At the bottom of the list, the total cost of the entire list is displayed.

Design

Main method pseudocode

Create initial List object
Declare int option = 0

While(int !=4)

- Prompt user with menu:
    - 1. Add item to grocery list
    - 2. Remove item from grocery list
    - 3. Print grocery list
    - 4. Exit
- If option <1 or >4 or not an int
    - Invalid entry, reenter option
- If option ==2 and List is empty
    - Invalid enter, reenter option
- If option == 1
    - Declare string name, unit
    - Declare int quantity
    - Declare double price
    - Prompt user: “Enter name of item:”
    - Prompt user: “Enter unit of item:”
    - Prompt user: “Enter quantity of item:”
        - If quantity <1 or not an int
            - Invalid entry, reenter
    - Prompt user: “Enter price of item:”
        - If price <0.0 or not a double
            - Invalid entry, reenter
    - Create item with parameters name, unit, quantity, price
    - Use overloaded “==” operator to check List object if new Item already exists
    - If Item does not already exist, add Item to List using addItem function
- If option ==2
    - Declare int num
    - Prompt user: “Select Item to delete from List:”
    - Print List
    - If selection is <1 or > than total items on list
        - Invalid entry, reenter
    - Delete item using deleteItem function
- If option == 3
    - If shopping list is empty, display “List is empty”
    - Print list

<u>List class methods</u>

Private Variables:

- Dynamically created array of Items
- Int listSize
- Int numItems

List()
Default constructor for List, sets initial listSize to 4 and creates new array with space for 4 items.
Sets initial number of items to 0.

~List()
Deletes itemArray to free allocated memory

addItem()
Adds item passed as parameter to List array
If number of items == listSize (array is full, need to create a new array with more space)

- Create temporary array and pass all items into new temporary array
- Increase listSize by 1
- Create new array of size listSize
- Pass Items from temporary array into new List array
- Delete temporary array
- Increment number of items

Else

- Add item to next open space in array (number of items)
- Increment number of items

deleteItem()
Deletes Item from list array based on int passed as parameter

- Create new temporary array and pass all items from the current List array to the new temporary array UP TO the int value passed as a parameter
- Skip the location on the current List array at the location of int and pass the rest of the Items to the temporary array (making sure to set the temporary array location -1 to not leave that space empty).
- Copy the contents of the new temporary array back to the original List array
- Delete temporary array

getCount()
Return number of items

getTotal()
Loops through array and adds the item totals to a total for the entire list
Returns total

print()
Loops through array:

- Prints name, unit, price, and quantity for each Item on the list.
- Also prints total cost for the item via getItemTotal() function

Prints total cost at the end of the list via getTotal()

operator==()
Item passed as parameter
Loop through array

- If Item name in List == item name in parameter:
  - Return true

Item class methods

Private Variables:

- String itemName, unit
- Int quantity
- Double price

Item()
Default constructor for Item, sets itemName and unit to " ", quantity to 0, price to 0.0.

Item(string, string, int, double)
User defined constructor for Item, sets all variables to those defined by user.

getItemUnit()
Returns unit.

getItemName()
Returns itemName.

getItemQuantity()
Returns quantity.

getItemPrice()
Returns price.

getItemTotal()
Returns quantity * price.

Testing

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Input too low	Input <1	main() while(option<1)	"Invalid selection, please select a number from 1 to 4"	"Invalid selection, please select a number from 1 to 4"
Input too high	Input>4	main() while(option<4)	"Invalid selection, please select a number from 1 to 4"	"Invalid selection, please select a number from 1 to 4"
Input not an int	Input “a”	main() while(!cin)	"Invalid selection, please select a number from 1 to 4"	"Invalid selection, please select a number from 1 to 4"
Input in correct range	Input ==1	main() calling *if* statement 1	calls *if* statement 1	calls *if* statement 1
Input in correct range but list empty	Input ==2	main() `if (option == 2 && shoppingList.getCount() == 0)`	"Invalid option. Your Grocery List is empty! Add something first!"	"Invalid option. Your Grocery List is empty! Add something first!"
Input item already on list	Input “Apples” when Apples already in list	main() `if (shoppingList == item)`	“This item is already on your list!”	“This item is already on your list!”
Input item information to list	Input name, unit, quantity, price	main() if(option == 1)	Displays name, unit, quantity, price, total price	Displays name, unit, quantity, price, total price
Input item name with 2 words	Input “Red Peppers”	main() if(option == 1) getline()	Item name correctly entered and displayed	Item name correctly entered and

			with print() function	displayed with print() function
Input item quantity <1	Input <1	main() if(option == 1)	"Invalid selection, please select a number greater than 1"	"Invalid selection, please select a number greater than 1"
Input option 3 "Print Grocery List"	Input option 3 "Print Grocery List"	main() if(option == 3)	Prints grocery list	Prints grocery list
Input option 2 "Delete Item from List"	Input option 2 "Delete Item from List"	main() if(option == 2)	Select item number to delete, then print new list without item	Select item number to delete, then print new list without item

Reflection

My initial design worked well but there were some changes I implemented in my final code.

During my first tests of the code, I kept getting memory leaks relating to my array initialization. I eventually realized I was not correctly deleting the temporary arrays I was using to transfer my List array into a new larger array when an Item object was being added. I corrected this by adding `delete[]tempArray;` to both my addItem() and deleteItem() functions to free that allocated memory. This seems to have resolved the memory leaks.

I ran into a problem with inputting a 2 or more word item name as well. I initially resolved this issue using the getline() function, but I then encountered an error in which keys were being stored in the buffer and skipping over my next inputs. This was resolved by using cin.ignore() for 100 instances of "\n". On a similar note, I struggled to find a satisfactory solution for efficient input validation that would valid non-integer inputs. I found !cin to be a great option as this is simple, easy to read, efficient, and seems to be very effective.

It took me a while to figure out how to add objects to my List array, I probably spent the most time of this project on this issue before even beginning my design phase. Once I figured out the key of creating new arrays with enough space for each extra Item object, the design became much more clear and I was able to generate a design and begin coding fairly quickly.

This was a great project for me to practice the use of dynamic arrays of objects and a great introduction to using overloaded operators. I have not used temporary arrays as temporary

storage for new arrays before, and it seems to be a very elegant solution to the problem of increasing array size.