**Assignment 2(Reflection Document)**

**Woohyuk Yang**

[**yangwo@oregonstate.edu**](mailto:yangwo@oregonstate.edu)

1. **Requirements:**

The requirement is to make a grocery shopping list program. I have to make two classes; one is Item class and the other is List class.

Firstly, Item class has to have data elements for item name, unit(i.e. can, box, pounds or ounces), number to buy, and unit price. And the program has to display extended price for each item in the end with total price using “number to buy” variable and “unit price” variable.

Secondly, List class needs to let each list objects store item objects. Whenever each item is newly added then an item object should be created to the list object. And the List class starts being able to store 4 item objects initially. When implementing this program, it is banned to use vector.

Once the classes are made, the program has to be able to perform few functions; creating lists and items, and removing items, and finally showing the shopping list for the user. The program has to prompt the user to enter the name, unit of sale, the number to buy, and the unit price when making lists and items. And in the end, it should be able to display each item in the list, the number of items, the unit of sale, the unit price, the extended price for each item, and in the end the total price for all items.

With the functions described above, the program has to validate whether the item newly input by the user already exists in the list. The program should overload the == operator to confirm that.

1. Class design on next page.
2. **Class Design**

|  |
| --- |
| Item |
| -name  -unit  -numToBuy  -price |
| +SetName() +GetName()  +SetUnit() +GetUnit()  +SetNumToBuy() +GetNumToBuy()  +SetPrice() +GetPrice()  + friend bool Operator==(Item a, string b) |

|  |
| --- |
| List |
| -Item\*\*(one for list)  -Item\*\*(one for temporary storage when the list needs to be resized)  -capacity  -emptyPos |
| +List() +~List()  +ShowMenu()  +AddToList()  +UserInput()  +RemoveFromList()  +DisplayList()  +CalTotalPrice() |

Item class is quite simple because its main purpose is to store information about each item itself. It stores name(string), unit(string), numToBuy(int), price(double), and each variable has its own accessor and mutator. The class “friend bool Operator==(Item a, string b)” to overload the operator. This lets the == operator to validate whether newly typed name(stored in string b) is same with the name already existing in other item objects. It is designed to return “true” if the newly type name is same with that of the other existing item object.

List class is about dealing with item objects. To add and remove item objects easily, I found out making the list of items as an array of pointers to objects not an array of objects merely. And to resize the array easily I assigned another double pointer variable to make a temporary array which will be used to resize the list. I designed List constructor and destructor to allocate the list and deallocate the list respectively. ShowMenu() lets the user choose which function user wants the program to perform. The menus are composed of 1)Add to list, 2)Remove from the list, 3)Display the list, 4)Quit the program. When user choose 1)Add to list then it calls function AddToList() inside ShowMenu(). This method applies to other 2 options and the last option quitting the program does not have extra function because it can be implemented simply inside a function. When it comes to AddToList() function when capacity is not enough, resizing the array is needed. The temporary array with bigger capacity is made . When enough place for another pointer to item object is left, it calls the function UserInput() which accepts variables and store them to each item objects using Set functions of each item objects. RemoveFromList() finds the exact place whose name is same with the user did input to erase. Then it deletes an item object the user wants to delete. DisplayList() shows the name, the unit of item, the number to buy, the price, and the extended price for the item. It also calls CalTotalPrice() to display the total price for the items in grocery list. CalTotalPrice() calculates the total price for the items in the list after some simple calculations.

1. **Test Plan**

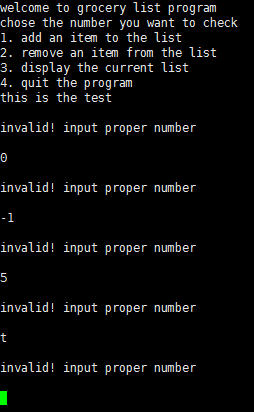
|  |  |  |  |
| --- | --- | --- | --- |
| **ShowMenu()** | | | |
| **1.AddToList** | **2.RemoveFromList** | **3.DisplayList** | **4.Quit** |

UserInput()

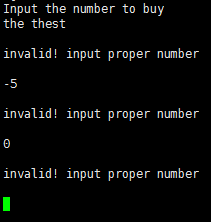
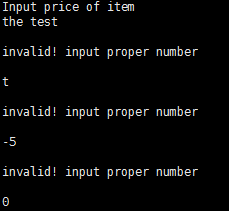
CalTotalPrice()

I illustrated the structure of the program to test concerned with the relations among functions.

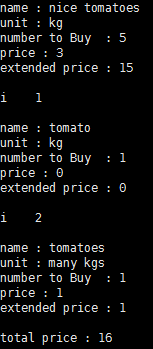
When the program starts the user will see some menus he/she can choose the number. The user is only supposed to input number because if they type characters or invalid number then the program could crash. So I put validating process. As the picture below it proves that it can distinguish numbers and characters.



So firstly when I input “1” and want to add an item to the list. When we need to deal with other numbers like when we need to input number for numToBuy variable and price variable. I checked out with different cases but it works well. But I let the price variable can accept 0 because an item could be free. But we do not add an item which we will not buy to a grocery list.

When the user input items’ name and unit, those variables are string type so it can accepts both numbers and characters.



Explanation)

Both cases for “nice tomatoes” and “tomato” work(string name)

Both cases for “kg” and “many kgs” work(string unit)

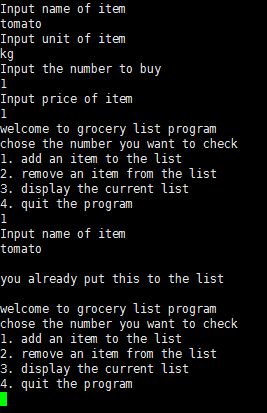
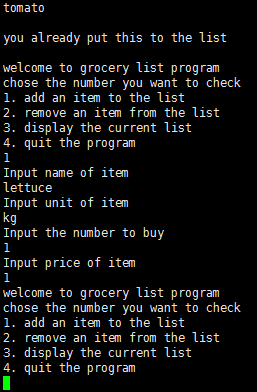
So all ShowMenu() part and UserInput() are done.

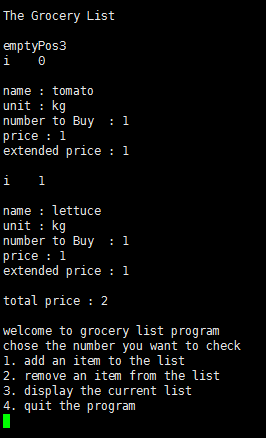
I tested both cases when I store more items than capacity and less items than capcity. To test easily I tested after setting default capacity to 2, Which is easier to test but same when it comes to rigorousness. And both cases worked for me. And both cases it worked well with displaying every elements and extended prices for each items and total price for the whole items. So AddToList() part is checked.

I tested RemoveFromList(). At first I removed the name that exists and checked it is erased with DisplayList() and tried to remove the name which does not exist. And made it return “there is nothing to delete.” message. I tested both cases for when the number of items is bigger than capacity and when the capacity is bigger than the number of items.

Testing AddToList() and RemoveFromList() functions, DisplayList() and CalTotalPrice() got tested automatically and those functions displayed right desired answers on the screen. The calculations for extended prices and total prices were accurate.

To finally test to check overloaded operator == works well. I stored tomato and I tried to add tomato again. And as you can see below it says “you already put this to the list”. And when you try to insert “lettuce” that is new to the list, it gets to be stored successfully.



As the picture on the left side, the list gets displayed well

4.reflection on the next page.

1. **Reflection**

I changed my design doing my implementation. At first I tried to put UserInput() function when Item constructor is invoked. But I designed the list constructor gets 4 initial basic pointers to item objects as an element so I changed UserInput( ) function to be located in List class so whenever AddToList() is called it can call UserInput() too.

As the requirement says the list starts with 4 items so I had to take care of that fact. At first it is empty and I can add 4 items without any difficulties but to store more than that I have to resize the array so I distinguished the cases. When I need to resize the array and when I do not need to.

Even when I implemented RemoveFromList() function, I approached to this as the same way to do AddToList() function. The difference is RemoveFromList() function requires resizing the array anyway. But there exists two cases like when the number of item is less than initial capacity(4 for this case). To keep the list with 4 pointers to item objects, I had to check with this case. And the other case is when capacity and the number of items stored in the list are same. At first I forgot to let user know that he/she is trying to delete the thing which does not exists. But I could deal with that easily after noticing that while my test.

DisplayList() function and CalTotalPrice() went both well as I designed first and there was nothing much to be concerned with.

The hardest part for me and I had to change a lot was overloading the ==operator. At first I tried to put that function inside List class. To do that I had to dynamically allocate another object to compare the name of that object with the others which already exist. That made sense at first but I found I had to delete the memory and reallocate that in the end to display the grocery list in the end. And it was wasteful so I changed the first plan and first implementation to enhance my efficiency. I made the operator to compare directly the name string the user would input with the names of the other objects already existing. With this better way I did not have to do deleting and reallocating the memory again.

This time with assignment2 , Every functions had their own features, I did not have to use my menu functions. But as more functions get added to this program, the necessity of using menu functions will grow certainly