# ­Fall 2016.

# **Grocery Store Inventory System**

File IO and Final assembly

IPC44 Project (Milestone 5, Dec. 2nd)

Structure of the project:  
In each milestone, a source file is provided under the name ipc\_msX.c that includes the main() tester program. (X means the milestone number from 1 to 6)

The main program acts like a tester (a unit test). I have Coded the functions in this file and have tested them one by one using the main function.

Milestone 1: ThE User Interface tools (Oct 29th)

In ipc\_ms1.c is about the following functions:

void welcome(void);

Prints the following line and goes to newline  
>---=== Grocery Inventory System ===---<

void prnTitle(void);

Prints the following two lines and goes to newline

>Row |SKU| Name | Price |Taxed| Qty | Min | Total |Atn<  
 >----+---+--------------------+--------+-----+-----+-----+------------|---<

void prnFooter(double gTotal);

Prints the following line and goes to newline  
 >--------------------------------------------------------+----------------<

Then if gTotal is greater than zero it will print this line: (assuming gTotal is 1234.57) and go to new line.  
> Grand Total: | 1234.57<

Use this format specifier for printing gTotal : **%12.2lf**

void clrKyb(void);

“clear Keyboard” Makes sure the keyboard is clear by reading from keyboard character by character until it reads a new line character.

void pause(void);

Pauses the execution of the application by printing a message and waiting for user to hit <ENTER>.

Print the following line and DO NOT go to newline:  
 >Press <ENTER> to continue...<  
 Then, call clrKyb function.

int getInt(void);

Gets a valid integer from the keyboard and receiving a result. If the integer is not valid it will print:  
"Invalid integer, please try again: "

and try again.

Flow chart for this function is below:



int getIntLimited(int lowerLimit, int upperLimit);

This function uses getInt() to receive a valid integer and returns it. This function makes sure the integer entered is within the limits required (between ***lowerLimit*** and ***upperLimit*** inclusive). If the integer is not within the limits, it will print:

> "Invalid value, *TheLowerLimmit* < value < *TheUpperLimit*: " <   
and try again. *(Change the lower and upper limit with their values.)*

Flow chart for this function:



double getDbl(void);

Works exactly like ***getInt()*** but scans a double instead of an integer with the following error message:   
"Invalid number, please try again: "

double getDblLimited(double lowerLimit, double upperLimit);

Works exactly like ***getIntLimited()*** but scans a double instead of an integer.

Milestone 2: The Application User Interface (5th November)  
Now that the user interface tools are created and tested, we are going to build the main skeleton of our application. This application will be a menu driven program and will work as follows:

1. When the program starts the title of the application is displayed.
2. Then a menu is displayed.
3. The user selects one of the options on the Menu.
4. Depending on the selection, the corresponding action will take place.
5. The Application will pause to attract the user’s attention
6. If the option selected is not Exit program, then the program will go back to option 2
7. If the option selected is Exit program, the program ends.

The above is essentially the pseudo code for any program that uses a menu driven user interface.

created three functions for 2nd stage of the project:

**int yes(void)**

Receives a single character from the user and then clears the keyboard (clrKyb()). If the character read is anything other than “Y”, “y”, “N” or “n”, it will print an error message as follows:  
**>**Only (Y)es or (N)o are acceptable: **<**and goes back to read a character until one of the above four characters is received.   
Then, it will return 1 if the entered character is either “y” or “Y”, otherwise it will return 0.



**int menu(void)**

Menu prints the following options:**><**  
**>**1- List all items**<**

**>**2- Search by SKU**<**

**>**3- Checkout an item**<**

**>**4- Stock an item**<**

**>**5- Add new item or update item**<**

**>**6- delete item**<**

**>**7- Search by name**<**

**>**0- Exit program**<**  
**>**> **<**

Then, it receives an integer between 0 and 7 inclusive and returns it. Menu will not accept any number less than 0 or greater than 7 (Use the proper UI function written in milestone 1).

**void GrocInvSys(void)**

This function is the heart of your application and runs the whole program.

GrocInvSys, first, displays the welcome message and skips a line and then displays the menu and receives the user’s selection.

If user selects 1, it displays:  
**>**List Items!**<** and goes to newline

If user selects 2, it displays:  
**>**Search Items!**<** and goes to newline

If user selects 3, it displays:  
**>**Checkout Item!**<** and goes to newline

If user selects 4, it displays:  
**>**Stock Item!**<** and goes to newline

If user selects 5, it displays:  
**>**Add/Update Item!**<** and goes to newline

If user selects 6, it displays:  
**>**Delete Item!**<** and goes to newline

If user selects 7, it displays:  
**>**Search by name!**<** and goes to newline

After receiving a number between 1 and 7, it will pause the application and goes back to display the menu.

If user selects 0, it displays:  
**>**Exit the program? (Y)es/(N)o): **<**and waits for the user to enter “Y”, ”y”, “N” or “n” for Yes or No.   
If the user replies Yes, it will end the program, otherwise it goes back to display the menu.



Milestone 3: The Item IO

Milestone 4: Item Storage and Retrieval in an array

Milestone 5: File IO and Final Assembly