# Online Store Project Assignment Four

## Introduction

Project Four introduces the ability to read and write data to files. This project will create an online store application circa 1990. Ideally, this assignment would be done using web pages and a browser user interface. But to keep things simple we’re going back in time to the glorious 1980s when Bulletin Board Systems (BBS) ruled the network and everyone used a 14400 baud dial up system (check out the [Wikipedia entry](http://en.wikipedia.org/wiki/Bulletin_board_system) for some history). In those days everything was displayed in basic text in a console. So the application you will build is a standard C++ console project (similar to the labs). The store will contain basic operations to shop for items, check out of the store, calculate taxes and totals.

All online store applications are “data driven” in that the contents of the store changes over time. In addition, shipping rates change over time. So this project will read shipping rates and tax rates in from a file. It assumes that these rates are based upon the zipcode provided by the user. The sample data for this file is located in the Canvas assignment.

When a consumer checks out of the store with their shopping cart, most eCommerce applications will email an invoice to the customer. Unfortunately, emailing receipts to customers is beyond the scope of this class! So you will need to write the invoice to a file. The format is described below. You will need to upload a sample output file along with the .cpp file to Canvas when turning in the assignment.

### Project Objective:

To gain experience with reading and writing files. Saving data into structures and creating arrays of structures.

## Steps and Requirements

Follow the steps below to create your program.

* Display a welcome statement to the user. Come up with a creative name for the store – such as “Hot Tub Time Machine Parts and Supplies”.
* The main() function should have a loop that displays a menu of shopping options. This is the control loop for the entire program. Once the user decides to check out or exit the store, the loop can stop.
* The main function will also need to define all of the variables to be passed to the functions listed below. This includes the
  + Customer name, address, city, zipcode
  + Shopping cart/catalog array – the simplest way to track a catalog of items is to define a structure which will contain the item description, the price of the item, the quantity purchased, and the weight of the item. The weight of items can be in ounces or grams (pick one). Then define an array of structures to track the item quantities.
* Create the following functions and place the function calls at the appropriate spots in the main() program.
  + loadShipping() - load the shipping rates and tax rates from the sample data file. This function should be called once at the beginning of main().
    - This function will receive an array of structures as input. You need to use a structure to save each rate for later use. Define the structure at the top of the program. There will be multiple lines in the file. The function does not have to return anything.
    - The layout of the file is
      * Zipcode taxrate USPSrate UPSrate FedexRate
    - Your function should validate the input data from the file. If data is invalid, skip over that line and proceed to the next line of the file. Things to check for include
      * Zipcode longer than 5 digits
      * Negative or invalid tax and shipping rate values. In project one, you hardcoded a shipping rate for each carrier. The file will now contain the rate (as a real number). When you look at the text file you will see some Los Angeles zipcodes with invalid data. Do NOT fix the data. Your code should handle these problems by displaying an error message and skipping that row of data.
      * The sample file contains all zip codes for **California**. This is over 2600 zip codes. Your array needs to be at least this big.
      * Hint – Make a copy of the zip code file and delete all lines after line 15. This will create a much smaller set of zip codes to test your program. Once you get everything working, use the main zip code list from Canvas.
  + displayMenu() –It displays the menu options. It does not need any input parameters. It returns the menu choice that the user selected. It should clear the screen each time it is called (hint – one option is to use system(“cls”)). It should validate the user input to make sure they have selected a valid option. The menu should have the following choices:
    - Create Customer Account
    - Shop for Items
    - Proceed to Check Out
    - Exit Store (without buying)
  + createCustomer() – This function prompts for the user’s name and address and saves the results. Input parameters include the name, address, city, and zip. If you choose to use string data types, then you will need to pass by reference. This function does not return anything.
  + maintainCart() – This function allows the user to add an item into their shopping cart. The shopping “cart” is represented by an array. This array is an input parameter to the cart function. Prompt the user with a catalog of items to purchase. You can make up your own list of items with prices that you want to sell (have at least 4-5 items). The user will choose a specific item. Validate the choice. Once a valid choice is made, then ask the user for the quantity to purchase. Save the quantity into the correct array slot. For example,
    - cart[0] – Hut Tub Cover, Price = $150.00
    - cart[1] – Pool Chemicals, Price = $12.00
    - cart[2] – Bubbles, Price = $18.00
    - cart[3] – Time Machine, Price = $4000.00
  + checkOut() – This function will “process” the customer order. In a true eCommerce application, you would charge a credit card and calculate shipping. For this application, the check out consists of calling the following functions. The input parameters are the zipcode, the quantity array. If the zipcode is empty (in the case where the user did not enter their customer info), prompt the user to enter a zipcode at this point.
    - displayReceipt() – This function should loop through all items in the cart and display the item, quantity, price, and extended amounts (price & quantity). This function returns the subtotal of all the items purchased.
    - calculateShipping() – Add up the total weight of all products ordered (quantity > 0). Prompt the user for the shipping method (USPS, UPS, or FEDEX). Based upon their zipcode, find the appropriate rate. Use the following table to calculate the shipping amount. Return this amount from the function.

|  |  |
| --- | --- |
| **Weight** | **Modifier** |
| 1-10 | 1x |
| 11-50 | 5x |
| > 50 | 10x |

For example, total weight = 50 ounces shipped via USPS to 90001, results in amount = 5 \* 1.55 = $7.55

* + - calculateTax() – This function should receive the location of the customer (zipcode) and the subtotal of the order as parameters. Look up the zipcode in the zipcode array to get the tax rate. Calculate the tax amount = subtotal \* rate. It should return the tax amount.
    - calculateTotal() – This function should receive the subtotal, shipping amount, and tax amount. It then returns the total. It should verify that the input amounts are valid.

After calling the above functions, the checkOut() routine should display the subtotal, tax, and total amounts to the screen. It should save the invoice or receipt to a file. You should print out the customer name at the top. Follow this with the item descriptions, price, quantity and extended price. Then print the subtotal, tax, shipping, and grand total (on separate lines of the file). Name the file “invoice.txt”.

* + Make sure not to print out any items that may not have been ordered (quantity = 0) from the cart.
* **Extra Credit One (2 points) –** Use the C++ date functions to print out the current date in the saveInvoice function. This can be tricky to get formatting correct.
* **Extra Credit Two (4 points)** – For the saveInvoice function, save the file in .html format. Use basic tags such as header <h1>, paragraph <p>, lists <ul> to format the invoice. With the HTML format, the grader should be able to double click on the file and have it launch to a browser.

## Submission Details

Turn in the following files:

* .cpp file (source code)
* Invoice.txt file (the file does not have to be named exactly this)
* EC – invoice.html file (optional)

You **must** follow all of the good programming practices discussed in class:

* Comment your code thoroughly.
* Indent your code appropriately.
* Use meaningful variable names.
* Do NOT use global variables.
* Provide the user with understandable prompts and instructions.
* Make sure your name is included in comments at the top of your code.
* You are NOT allowed to use goto statements in this or any other COMP 51 projects.
* …

If this is not done, points will be deducted from your program and it will be impossible to earn an ‘A’.