Project 3 – Daily Ledger

Overview

Once the classes for solving a problem have been correctly identified and designed, multiple objects of any class type can be created. It is also possible to have arrays of objects. In these objects, it is possible to use any data member field as the key and sort the object array on that key according to the required order. In this project, you are required to create arrays of objects and develop a solution to sort them using a variety of criteria.

Learning Objectives

The focus of this assignment is on the following learning objectives:

- Be able to identify classes and objects from a problem description
- Be able to identify class functionality
- Be able to identify an incremental approach to developing the program solution
- Be able to implement the program solution based on the identified approach
- Understand how the program solution can evolve over time

Prerequisites

To complete this project, you need to make sure that you have the following:

- C++ and the g++ compiler
- A C++ IDE or text editor (multiple editors exist for developers)
- An understanding of the material presented in class.
- An understanding of the material covered in ZyBooks.

Problem Description

You are to implement a program for creating a daily transaction ledger for a shop. For this project, you will need to utilize the concepts of arrays and object creation that have been discussed in class. The list of requirements and constraints for the system are as follows:

1. The system must be able to maintain records of all transactions done on a date. Each transaction will occur at a given time, identified by the hour, minute, and second. In addition, each transaction will have a unique ID (a number) for each transaction that will be randomly generated by the system. You can assume no more than 10 transactions will happen on a single date. You must ensure that each transaction ID is unique. You can find more information on generating random numbers by going to http://www.cplusplus.com/reference/cstdlib/rand/.
2. For each transaction, an item will be sold. This item will have a name, a cost price, and a selling price, with the selling price greater than the cost price for a profit (selling price – cost price). Each item should have a different selling price and profit.
3. It is possible that a transaction will be voided in which case that transaction should be deleted from the system. If the maximum of 10 transactions does not exist in the system, then more transactions should be possible for the day.

4. Each day a summary of transactions will be printed. The summary will include for each transaction, the time of the transaction, the transaction ID, the name of the item, the cost price of the item, the selling price of the item and the profit.
5. You will need to create a console menu for using the system. This menu should allow you to create a new transaction, void a transaction, and print a summary of transactions. To create a new transaction, the user will be prompted to enter the hour, minute, and second of the transaction, and then to enter the details for the item being sold. To void a transaction, the user should enter the transaction's ID (obtainable through the summary discussed next). The summary will include for each transaction, the time of the transaction, the transaction ID, the name of the item, the cost price of the item, the selling price of the item and the profit. In addition, after printing the transactions, it will print the total profit for the day and the average profit for each item.
6. After selecting to print the summary, you will offer a menu of choices for the order in which transactions are printed in the summary. The choices will allow the user to choose to sort the transactions by transaction ID, by item names (alphabetically), or by increasing order of profit. You can choose any of the sorting algorithms described in the class for this.

It is recommended that in order to test your system, you create some test data for transactions so that you do not need to reenter data each time you run the system. However, you should not include this test data in the final submission.

Tasks

For this project, you must complete the following tasks in the order that they are described:

1. From the problem description, create a list of all classes that you can identify. For each class, list the associated data member variables and identify an initial set of member functions.
2. List out a set of steps that you will take to implement your solution to the problem. Each step refers to an increment of the program that you will be creating. It is recommended to complete the implementation of a single logical action per step.
3. Begin implementing your program by using the plan that you created in step 2. For each step, save a snapshot of your program design once that step is completed. Each snapshot should be saved to its own directory so that you can go back to any version directory, should you encounter problems.
4. Once you have finished implementing your solution, reflect on the process that you followed. Did you wind up with the same classes as you initially identified? Did you need to change any of the functionality or add unexpected details? Did you have to deviate from your plan? Write a description of any details that needed to change as you worked on your solution.

Your responses to tasks 1, 2, and 4 should be submitted as a word document called Answers. Each snapshot should be saved to a directory labelled with a names of the form "Step#" where # refers to the step from your plan. One of these should be your final implementation.

Grading Breakdown

[10 pts] Text menu

[5 pts] Manage multiple transactions correctly and efficiently

[8 pts] Add a transaction

[5 pts] Correctly calculates profit

[5 pts] Generate random id correctly

[8 pts] Remove a transaction

[20 pts] Sorting algorithms: by id, item name, and profit

[10 pts] Print summary: formatting, data

[10 pts] Sufficient and clear class divisions

[5 pts] Clear, easy-to-read code

[8 pts] Snapshots of previous *working* iterations, including descriptions of changes from the previous iteration

[6 pts] Reflection of the process

Submission

Points will be deducted for not following these instructions. Before submitting this project in eLearning make sure that you follow the following steps:

1. Make sure that your name appears at the top of each file. This should be done as a comment for any source code files.
2. Copy each snapshot directory and your Responses document into a directory called “Project 3”. Do not include any additional files. Zip up this directory into a .zip (points will be deducted for using any other compressed format).

Turn your zipped up project into eLearning.