CS 162 -- Winter 2009

Homework Assignment #1

Due Date: Thursday, February 5th, 2009

Goal: To review C++ concepts and write a C++ program using functions, Multi-dimensional arrays (you can use struct as long as you use at least 1 Multi-dimensional array), strings and files, and basic search function.

Requirements and Problem Statement: Write a Functional Specification and a C++ Program that implements a basic banking software database application.

The database is a simple in-memory database implemented using Multi-dimensional arrays (we will later modify the database to use more advanced structures, but let us start simple)

The banking software should have a:

1) Phase 1: Simple Command Line Interface
2) Phase 2: File Interface
3) Phase 3: Searching for a Record using Simple Sequential Search (from class)

The Command Line and File Interfaces should have the following sub-interfaces:

a) Customer Interface (this should be simple like an ATM machine interface):
    a. Functions: Balance Inquiry
    b. Deposit Funds
    c. Withdraw funds
    d. Transfer Funds to another account (like from Checking to Savings or to Credit Card)
    e. Anything else that you want to add
    f. Quit the application

b) Bank Employee/Teller Interface:
    a. Everything in the Customer Interface, plus
    b. Interface to ADD a new Customer to the Bank
    c. Interface to DELETE a Customer from the Bank
    d. Search for a Record based on a "Name of a Customer" (using simple Sequential Search)

c) Bank Supervisor/Manager Interface:
    a. Everything in Bank Teller Interface, plus
    b. Ability to List Total Number of Customers in Bank
    c. Ability to List Total Amount of Money in the Bank
    d. Ability to List Total Deposits in a Day
    e. Ability to List Total Withdrawals in a Day

f. Ability to see/query detailed LOG of all transactions

Example 1.0 Functional Specification (you are expected to modify)

This document describes version 1.0 of the Banking Database Application

Components

Banking Database consists of these components:

- a simple in-memory database of Banking Records
- a simple command-line an file user interface allowing the user to:
- Customer Interface (this should be simple like an ATM machine interface):
  - Functions: Balance Inquiry
  - Deposit Funds
  - Withdraw funds
  - Transfer Funds to another account
  - Anything else that you want to add
  - Quit the application
- Bank Employee/Teller Interface:
  - Everything in the Customer Interface, plus
  - Interface to ADD a new Customer to the Bank
  - Interface to DELETE a Customer from the Bank
  - Search for a Record based on a "Name of a Customer" (using simple Sequential Search)
- Bank Supervisor/Manager Interface:
  - Everything in Bank Teller Interface, plus
  - Ability to List Total Number of Customers in Bank
  - Ability to List Total Amount of Money in the Bank
  - Ability to List Total Deposits in a Day
  - Ability to List Total Withdrawals in a Day
  - Ability to see/query detailed LOG of all transactions

Bank Database

Each record in the database has the following data associated with it:

```
char        name[SIZE];        // name of the customer
float       amount;            // amount
```

```
AccountType type;               // Type of Account, see enum
…. (etc)
```

"AccountType" is the name of a C++ enumerated type that should be defined as follows:

```
enum AccountType {
     unknown = -1, Checking, Savings, CreditCard, Instant Access
     };
```

Bank Account database is stored in a linear, indexed sequence. The first item in the database has index **0** (zero), the second has index **1** (one), etc. A database with N customers in it will use index numbers in the range **[0 .. (n-1)]**.

Bank Database Implementation

Implement your database as an multiple dimensional array of **Bank Customer Record** (you may optionally use struct, as long as there is at least one Multi-dimensional array).

No global variable is allowed. Global constants will be fine though.

User Interface Details

A document of this length cannot possibly specify every single detail of a user interface. The Student has the liberty to make assumptions and clearly state them in their implementation. There should be a "Known Issues and Limitations" section for Known Issues and Limitations with your program.

Documentation: Part of your evaluation of this program will include how you document your program (algorithm, flowchart etc).

Email your assignment (as text file or attachments of MS word) :

1. An algorithm written in pseudo code of the flow of logic used to solve this problem
2. A copy of your well documented source program
3. Include Known Issues and Limitations Section
4. Sample Program Input and Output (screenshots)
5. Your program (including the .EXE and intermediate files)

Reminder: Your instructor will compile, link and run your program as part of your evaluation.