PROJECT 2 PROGRAMMER MANUAL

MB. Katumba

CS121 – November 30, 2023

1. Problem Description

The program will compute and print out the monthly report of customers bank account activities. The ending balance will depend on the amounts deposited or withdrawn. The report should show how many withdrawals occurred and the total amount of withdrawals and show how many deposits and the total amount of deposits.

2. Data types and Classes

This program used one user defined data type, a struct and other primitive predefined data types in C++ language. The following subsections enumerate the data types used.

2.1 fstream:

Variables AccountFile, TransactionFile, ReportFile – streams to be connected to the input Output files.

2.2 string:

Variables accountFile, transactionFile, reportFile – pathname to the files

2.3 const int:

Variable MAX_ACCOUNTS – maximum number of accounts of data type struct Account.

2.4 double:

Variable totalDeposits, totalWithdrawals, beginningBalance, amount – account activities.

2.5 int:

Variables numDeposits, numWithdrawals, i, numAccount – account number, deposits count, withdrawals count, loop index.

2.6 char:

Variable transaction – type of transaction, deposits or withdrawals represented by letter ‘d’ or ‘w’.

2.7 struct:

Variable accounts – array accounts of type struct Account.

3. Files separation

3.1 Header file

banking.h -> contains libraries, struct and function definitions.

3.2 functions

cs121_P2.cpp -> contains functions implementations.

3.3 client (main)

Client.cpp -> contains the main program implementation.

4. High Level Program Solution

4.1 Algorithms:

Main program

1. Prompt user to the Account file
2. Call function getfileName to take the path entered by user and save it as string accountfilename.
3. Call function openFile to open data file, connecting it to the stream variable AccountFile.
4. Read the first entry of the accountfile which is the number of entries.
5. Print the value of the first entry to the console.
6. Check if the number of entries is a non-null positive integer.
    - If the number of entries is negative or a not an integer

Print Invalid number of entries on the console.

7. If the number of entries is valid, create an array of type struct Account of size 100.
8. Using a for loop, write the content of accountFile in the array of type struct and save in the variable element account number and name.
9. Close the accountFile.
10. Print the value saved in the array.
11. Prompt the user to enter a valid path of the monthly transactions.
12. Open the transaction file.
13. Read from the transactionFile, the account number and the beginning balance.
14. Update the value of the element beginning balance of the struct Account with the value read from transactionFile saved in the beginning balance.
15. While there is more to read in the transactionFile
    - 15.1 check the next character in the transactionFile, if the character is a newline, break.
    - 15.2 Check if the character is either a whitespace or a tab, ignore those characters.
    - 15.3 Read the next character from the transactionFile, and save it in the variable transaction of type char.
    - 15.4 Check if the account number saved in the array matches the account number read from the transactionFile.
    - 15.5 Else print on the console that the format of the transactionFile is wrong.
    - 15.6 Read the next value from the transactionFile and save in the variable amount.
      - 15.6.1 check the first character from the following element read, if the character is ‘d’ update the element account balance by adding and update the element total deposit and number of Deposits.

15.6.2 Check if that first character is ‘w’, if it is true, update element account balance by subtracting, update the element total withdrawals and number of withdrawals.

15.7 Compute the balance ending which is account balance plus beginning balance.

15.8 Print the report on the console, for each loop.

15.9 Prompt the user to input the complete path of the file where the report should be written.

15.10 Open the file and write the full report.

15.11 Close the file.

15.12 Print the file was successfully generated.

5. Limitations and Suggestions.

a. If the number of entries in the accountfile is mistakenly not a positive integer, the program will crash. Therefore, we could use a way to continue reading the file despite that fault.

b. The account number must be an integer for the program to work, in case account number contains letters, the program would crash, we could use a string to solve that.

c. The program can only process a database of 100 customers max, this is a personal choice , an improvement can be made.