Midterm 2024

> Spring semester
>
> 2023/24 Date: March 20, 2024

General Info

We have transaction history and list of person bank accounts (transactions_accounts.csv)

Account

- id: string — uuid[1] account id
- name: string — person name;
- validity_timestamp: uint[2] — account validity datetime or last datetime when person can execute operations: deposit, transfer (send and accept) and withdraw money.

Transaction

- id: string — uuid[1] transaction id
- timestamp: uint — timestamp transaction
- type: string — one of {deposit, transfer, withdraw}
- from: string — uuid account id for {transfer, withdraw} and None for {deposit}
- to: string — uuid account id for {deposit, transfer} and None for {withdraw}
- amount: double — amount of money

Transaction types:

- `deposit` money to the account (`from` field always empty). Then you make deposit the total money amount of incoming account is increasing;
- `transfer` money from one account to another (`from` could be equal `to`). Then you make transfer the total money amount from `from` account is decreasing and `to` account is increasing;
- `withdraw` money from the account (`to` field always empty). Then you make withdraw the total money amount of account is decreasing.

Code description and structure of project

You have `main.cpp` file with tests for tasks, you can use this file to contest your solution.

You have `bank_account.h` header file with needed classes, structure and functions. You should send this file with your code to the contest.

Tasks

Task 1

Implement Account constructor.

Task 2

Implement `operator<` overloading for Transaction structure according to transaction timestamp. If timestamps are equal, then you should compare by transaction type: **deposit < withdraw < transfer**.

Task 3

Implement `addTransaction` method for Account class which allows you to extend `Account::transactions` property. You don't need to look at corner cases such as account balance is lower then 0;

Task 4

Implement `getBalance` method, input parameter is datetime. You should return the balance for a specific date.

Probably, you could implement own comparator or overload `operation<=` for `Transaction`.

Task 5

Implement transaction (`readTransactions` function) and account (`readAccounts` function) reading from the file and then fill accounts by transactions (`fillAccounts` function).

Important: TransactionContainer cannot store similar within operator< meaning transactions by datetime and transaction type even for different users.

1. A **Universally Unique Identifier (UUID)** is a 128-bit label used for information in computer systems. UUID consists of 32 characters and numbers separated by dash sign (8-4-4-4-12 characters respectively). ↩ ↩

2. The unix time stamp is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. ↩