# Proiect de atestat Leonte Denis – Sistem Bancar

1. **Tema proiectului.................................................................................2**
2. **Functiile de sistem...............................................................................3**
3. **Clasa account......................................................................................3**
4. **Functiile i/o.........................................................................................3**
5. **Funcțiile clasei account........................................................................5**
6. **Login...................................................................................................5**
7. **Loop......................................................................................................**

**1.Tema proiectului**

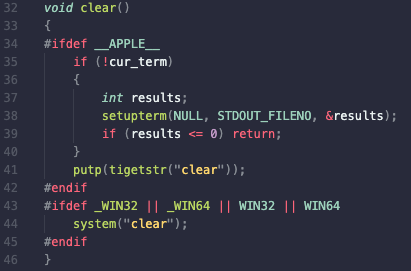
Tema proiectului este simularea unui sistem bancar, cu același tipuri de operații și tranzacții. Proiectul este focusat pe programarea orientată spre obiecte (POO) și folosește multe noțiuni ale acesteia. Ceea mai relevantă noțiune fiind clasele. Limbajul de programare ales este C++, fiind cel mai rapid, eficient și simplu din punctul meu de vedere.

Tema va lua în calcul niște probleme de nivel mediu, dar am evitat generarea unui număr de card, IBAN si CVC/CVV2. Ceea mai grea problemă întălnită a fost funcția de sistem writeToFile(account x), care a dat probleme de la bunul început.

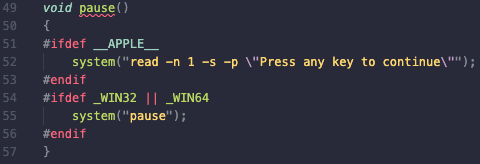
Sistemul încă nu este perfect, trebuie pus cazul înregistrării a 2 persoane cu același nume, creearea mai multor conturi pe o persoană, etc...

**2. Funcțiile de sistem**

Am deciz sa declar 2 funcții de sistem generalizate, din cauza că programez din mai multe sisteme operative, astfel încât sa am cate mai puține conflicte.

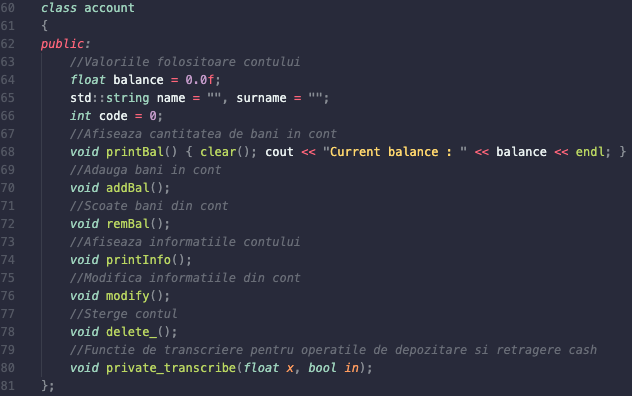
clear() - funcție care va șterge tot ceea ce este pe ecran.

pause() - funcție care va pune în pauză programul. Folosită de obicei la finalul funcțiilor care afișează valori.



**3. Clasa account**

Clasa account este clasa principala a programului. Nu voi merge în detalii la explicat ce este o clasa dar în mare este un obiect care vine cu un set de parametre și funcții predefinite de programatorul. Clasa account vine cu 4 variabile : float balance, care reprezinta valoarea din cont, string name și surname care sunt numele si prenumele, int code care este codul de referință a contului. Clasa vine și cu 7 funcții care vor fii explicate mai îndată.



**4. Funcțiile I/O**

Avem în total 4 funcții input/output.

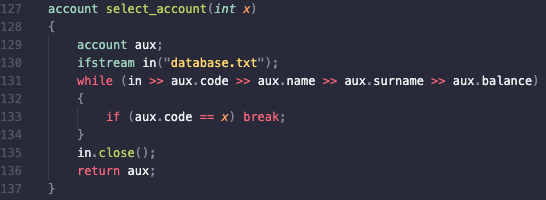
**void writeToFile**(account x, bool add) - funcția ceea mai importantă, poate fii folosita doar pe post de output.

bool add daca este adevarat defineste operația de adăugare în fișier a contului x, iar altfel va modifica contul x în fișier.

Începem prin declararea unui vector de conturi, citim fișierul și dacă add este fals, odata ce codul citit din fișier este egal cu codul din x, vom pune x in vector în loc la ceea ce este în fișier. Daca add este adevărat, atunci la finalul citirii din fișier vom adauga x în vector. Urmează sortarea fișierului cu metoda bubble sort iar la final vom scrie totul inapoi în fișier.

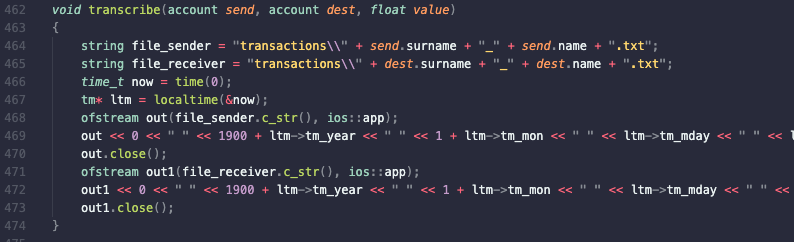


**account select\_account**(int x) - funcția de actualizare a contului, este folosită numai ca input. Declaram un cont auxiliar și citim valori din fișier pănâ când nu găsim un cod egal cu x. Atunci vom returna contul auxiliar.

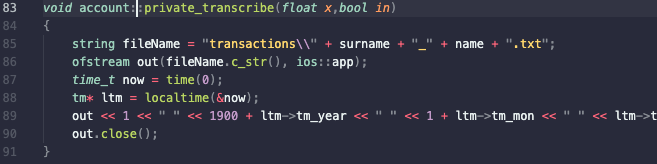
****

**void transcribe**(account send, account dest, float value) - obiectivul funcției este generarea unui fișier cu toate tranzacțiile făcute de user. Funcție este numai output.

Prima dată vom creea 2 string-uri ”transactions/prenume\_nume.txt”, unul pentru trimițătorul și altul pentru destinatar. Vom folosi niște funcții definite în **ctime** pentru a afla data și ora curentă iar în fișierul respectiv vor fii toate informațiile necesare citirii.



**void account::private\_transcribe**(float x, bool in) - functie clasei account care are același rol ca transcribe() numai că în loc sa fie tranzacția între 2 conturi este o tranzacție de cash-in și cash-out care este determinată de bool in.



**void changePass**(account x) - funcția are ca obiectiv schimbarea parolei. Înainte de a continua cu modificarea parolei, trebuie definit concept de parola. În acest contest, parola este criptată în SHA256, care este un algoritm imposibil de decriptat. Verificarea parolei este procesul de inserarea parolei curente, criptarii ei și compararea cu parola originală din fișier.

Vom verifica parola, insera parola nouă, cripta parola noua și înlocui parola veche din fișier cu parola nouă.





**void account::modify**() - functia clasei account care modifică informațiile de genul nume – prenume. Va cere parola, o va verifica și va schimba valorile ÎN CLASA.

**void account::delete**() - funcția clasei account care este menită să șteargă contul. Prima dată se va cere numele, prenumele și parola. Odată ce userul a dat confirmarea, va începe un procedeu asemănător cu cel in writeToFile() cu unica diferență că odată ce am ajuns la codul contului, NU vom adăuga contul în fișier, astfel ”ștergând” contul. Vom proceda la fel și cu fișierul ”accounts.txt”.

**void check\_history**(account x) - funcție care va afișa toate tranzacțiile efectuate.

Va începe cu citirea fișierului personal de tranzacții, iar va continua cu formatarea datelor din fișier astfel încăt să fie ușor de citit de user.

**5. Funcțiile clasei account**

Clasa account are 7 funcții, din care 3 au fost deja explicate.

**void printBal**() - afișează valoarea curentă .

**void addBal**() - va depune numerar în cont. Valoarea trebuie sa fie strict pozitivă.

**void remBal**() - va retrage numerar din cont. Valoarea trebuie sa aparțină intervalului (0, valoare din cont).

**void printInfo**() - va afișa informațile din cont(nume prenume valoare etc).

**6. Login**

La începutul programului userul este pus să aleagă între logare și înregistrare.

**Înregistrare**

Procedeul de înregistrare constă în verificarea codului maxim existent, inserarea datelor, verificarea datelor pentru a găsi un duplicat, introducere parolei de 2 ori pentru corectitudine, și inserarea datelor în fișier. Funcția de înregistrare este definită ca **void create\_user**().

Pentru a găsi codul maxim, vom deschide fișierul ”accounts.txt”, vom merge pe fiecare rând în căutarea unui cod maxim. O dată ce am găsit codul maxim programul va adăuga +1 la el astfel devenind cel mai mare. Continuăm cu introducera datelor, căutarea în fișier pentru un duplicat numelui și vom continua cu verificarea parolei. Odată ce toate datele au fost inserate, vom crypta parola, actualiza fișierul cu parolele ”accounts.txt” cu codul și parola, initializa clasa și vom apela writeToFile() să actualizăm fișierul ”database.txt”.

**Logare**

Procedeul de logare este asemănator cu înregistrarea. El constă în inserarea datelor de logare, verificarea numelui și prenumelui, inserarea și verificarea parolei. Funcția va returna true atunci când inserăm informațiile corecte și false în caz contrar. Funcția este declarată ca **bool login**().

Funcția va începe cu input-ul userului, verificarea numelui și prenumelui în fișierul ”database.txt”, daca există vom trece la verificarea parolei, dacă parola este corectă atunci vom modifica variabila globala **sel\_user** si o vom schimba cu codul contului. Această valoare ne va fii folositoare atunci cănd actualizăm contul.

**void login\_screen**() – funcție de meniu a ecranului de login.

Este un meniu cu 3 alegeri : login, înregistrare și ieșire.

**7.Loop**

Subprogramul care coordonează totul este **void loop**(). Prima dată va actualiza contul, șterge ecranul, afișa meniul, selecta o valoare din meniu și prin un switch vom apela la funcțiile folositoare.