**Documentatie – Project 2**

Initial, am declarat structurile care vor retine informatiile esentiale cu care voi lucre pe tot parcursul aplicatiei: **struct clients** : retine informatii despre un client(user) -> numele sau, numarul de tipuri de cont bancar avute, tipurile efective de cont bancar sub forma altei structure, numarul de clienti pe care ii are si informatii despre clientii respectivi, tot sub forma unei alte structuri**; struct financial\_account**: retine informatii despre un tip de cont bancar -> numele tipului (poate fi: ‘checking account’, ‘savings account’ sau ‘credit account’), balanta curenta (double), extrasul de cont (un sir de caractere ce retine toate operatiile indreptate inspre acest tip de cont bancar), registrul de tranzactii (un sir de caractere ce retine depozitarile, extragerile si transferurile ce au legatura cu acest tip de cont bancar) si raportul de cheltuieli (un sir de caractere ce retine platile efectuate dinspre si inspre useri-firma); **struct manage\_custumers**: retine informatii despre clientii unui user-firma -> nume client, numarul de plati efectuate de client firmei si un mesaj pe care userul il poate asocia clientului, in functie de propriile reguli (spre exemplu, raportat la numarul de plati efectuate). In plus, vom defini global, pe langa altele, un sir mare de caractere (**activity\_log**) in care vom retine toate operatiile efectuate de toti userii, cu ajutorul functiei pe siruri de caractere -> **strcat**. Tot cu ajutorul acestei functii pe siruri se va crea, din aproape in aproape, in cadrul fiecarei functii ce se ocupa cu o functionalitate legata de tranzactii, si sirurile: **transaction\_register** si **expense\_rep**. Mai mult, in cazul sumelor de bani, se va folosi tipul de date **double** (numere reale), fiind utilizate doar **primele** **2 zecimale**. Va fi nevoie, in unele momente, de transformarea unui numar in sir de caractere. Aici am folosit functia **sprintf**!

Alte variabile globale importante: **bin** (sir de caractere ce retine textele reziduale din buffer), **pass** (sir de caractere ce contine parola pe care trebuie sa o introduca userii pentru a putea avea acces la datele financiare), **aux** (litera ce extrage caracterul new line -> ‘\n’ de la finalul liniilor din buffer), **cmd** (retine cifra corespunzatoare operatiei dorite de user a se efectua), **nr\_clients** (retine numarul curent de clienti inregistrati), **p** (integer folosit pentru a citi numai date corecte din consola).

Urmeaza mai multe functii ce cuprind toate meniurile oferite user-ului.

Functia **security**() -> se ocupa cu verificarea corectitudinii parolei introduse de user. Acesta poate accesa datele financiare numai daca introduce parola corecta (in acest caz: ‘password’) sau poate inchide consola, scriind ‘x’. In cazul introducerii unei parole gresite, user-ul e atentionat si i se ofera posibilitati infinite de introducere a unei noi parole.

Functia **validate\_accountType** -> primeste pointer de tip sir de caractere (account\_type) prin care e salvat tipul de cont bancar introdus de user. Daca acesta este unul din cele 3 posibile, functia returneaza 1, altfel 0.

Functia **create\_account** -> creeaza un cont nou asociat numelui userului (user\_name) si un tip de cont bancar (account\_type) daca pana la acel moment user\_name nu a mai fost inregistrat in baza de date sau adauga un nou tip de cont bancar daca acel user\_name are deja cel putin un tip de cont bancar inregistrat anterior. Daca user\_name are deja inregistrat tipul de cont bancar (account\_type), se va iesi din functie, user-ul fiind atentionat ca nu poate inregistra de 2 ori acelasi tip de cont bancar. **Alocarea memoriei** pentru fiecare user si respectiv client are loc **dinamic**, din aproape in aproape (la fiecare noua inregistrare are loc un **realloc**, de size cu 1 mai mare decat marimea anterioara)

Functia **edit\_account** -> schimba numele user-ului cu numele user\_name in new\_user\_name, daca acest user e deja inregistrat in baza de date. Daca nu, va fi afisat un mesaj corespunzator

Functia **delete\_account** -> sterge tipul de cont bancar (account\_type) asociat user-ului cu numele: user\_name, daca acesta exista. User-ul, in schimb, nu este sters, el exista in continuare in baza de date, chiar daca e posibil sa ramana fara niciun tip de cont bancar. Daca nu exista in baza de date numele user-ului sau tipul de cont transmise functiei, vor fi afisare mesaje corespunzatoare.

Functia **verify\_balance** -> verifica daca balanta unui anumit tip de cont bancar ramane pozitiva dupa eventuala modificare cauzata de o tranzactie

Functia **deposit** -> depoziteaza in contul: account\_type, al lui user\_name, suma de bani – val (double) atata timp cat exista inregistrat numele clientului si acesta are asociat tipul de cont bancar transmis functiei, iar suma de bani nu produce o balanta negativa. In caz contrar, se va afisa un mesaj corespunzator conditiei neindeplinite. In functie de valoarea numarului val, functia deposit indeplineste fie functionalitatea de depozitare, fie de extragere a unei sume de bani.

Functia **tranfer** -> transfera suma de bani – val (double) – din contul: account\_type al user-ului cu numele: user\_name, in contul: second\_account\_type al user-ului cu numele: second\_user\_name. Daca transferul nu poate avea loc din motive identice cu cele mentionate anterior, se va afisa un mesaj corespunzator care sa clarifice acest lucru.

Functia **pay** -> realizeaza, ca functia anterioara (transfer), un transfer de bani de la un client la altul, cu precizarea ca celui de-al doilea user i se va incrementa numarul de clienti – second\_user\_name efectueaza pentru prima data o plata catre user\_name (second\_user\_name devine deci o asa-zisa ‘firma’), clientului user\_name i se va incrementa numarul de plati efectuate user-ului second\_user\_name si, dupa caz, se va intializa numele clientului cu user\_name (daca pasul acesta nu a fost realizat inainte). Daca transferul nu poate avea loc din motive identice cu cele mentionate anterior, se va afisa un mesaj corespunzator care sa clarifice acest lucru.

Functia **view\_clients** -> afiseaza toti clientii user-ului cu numele: user\_name sau afiseaza un mesaj corespunzator in cazul in care ori user\_name nu are niciun client, ori nu exista inregistrat numele: user\_name in baza de date.

Functia **add\_info** -> realizeaza salvarea unui mesaj - info, introdus de un user-firma cu indexul: index, indreptat inspre unul din clientii sai, cu indexul: id\_client. El poate fi modificat/ inlocuit oricand. Daca unul din indecsi nu exista, se va afisa un mesaj corespunzator.

Functia **view\_account\_statement** -> afiseaza extrasul de cont pentru user-ul cu numele: user\_name si tipul de cont bancar: account\_type (adica afiseaza toate depozitarile, extragerile, transferurile si platile efectuate care au legatura cu acest tip de cont bancar). Daca nu e inregistrat: user\_name-ul ori account\_type\_ul, se va afisa un mesaj corespunzator.

Functia **view\_balance** -> afiseaza balanta curenta pentru tipul de cont bancar: account\_type, asociata user-ului cu numele: user\_name. Daca nu e inregistrat: user\_name-ul ori account\_type\_ul, se va afisa un mesaj corespunzator.

Functia **view\_transaction\_register** -> afiseaza, pentru un tip de cont bancar: account\_type, asociat unui user cu numele: user\_name, toate depozitarile, extragerile si transferurile ce au legatura cu acest tip de cont bancar. Daca nu e inregistrat: user\_name-ul ori account\_type\_ul, se va afisa un mesaj corespunzator, acelasi lucru avand loc si daca, la momentul apelarii functiei, nu e inregistrata nicio depozitare/extragere/ niciun transfer pe acest tip de cont bancar.

Functia **view\_expense\_rep** -> afiseaza, pentru un tip de cont bancar: account\_type, asociat unui user cu numele: user\_name, toate cheltuielile (platile efectuate catre useri-firme) ce au loc dinspre tipul de cont bancar: account\_type. Daca nu e inregistrat: user\_name-ul ori account\_type\_ul, se va afisa un mesaj corespunzator, acelasi lucru avand loc si daca, la momentul apelarii functiei, nu e inregistrata nicio cheltuiala pe acest tip de cont bancar.

Functia **view\_activity\_log** -> afiseaza toate operatiile efectuate pana in momentul apelarii functiei, operatii realizate de catre toti userii, inca de la inceputul rularii programului (toate aceste operatii sunt salvate in sirul de caractere activity\_log).

Functia **export\_data** -> exporta datele financiare (pentru fiecare user: numele sau, balanta tuturor tipurilor de conturi bancare – daca un user nu are un anumit tip de cont bancar, in coloana respectiva va aparea mesajul: ‘doesn’t exist’ – si numarul de clienti – poate fi si 0) catre un fisier cu denumirea – file\_name (contine extensia .csv sau .txt, in functie de preferinta user-ului). De fapt, referirea la fisierul cu denumirea ‘file\_name’ are loc cu ajutorul unui **pointer** definit la inceputul lucrului cu fisierul. (FILE \*file = fopen(file\_name, “w”) ). Daca nu a putut avea loc deschiderea cu succes a fisierului, se va afisa o eroare prin functie ‘perror’ si se va iesi din functie. Cautarea fiecarui tip de cont bancar are loc prin intermediul unui vector de cuvinte ce retine numele acestor tipuri de cont bancar (3 cuvinte, un cuvant avand maxim 19 litere + caracterul nul) -> acc[3][20]. La finalul lucrului cu fisieirul, acesta se va inchide.

La nivelul functiei **main():**

Un nou user trebuie sa introduca mereu parola pentru a putea accesa datele financiare. Lui ii vor fi prezentate , cu ajutorul meniurilor, toate functionalitatile pe care le are la dispozitie, tot ce trebuie el sa faca fiind sa introduca **cifra** corespunzatoare operatiei dorite, cifra ce apare in partea stanga a descrierii functionalitatilor. In marea majoritate a cazurilor, userul nu poate iesi din meniu pana nu introduce **o cifra valida**. Ori de cate ori introduce litere sau o combinatie de litere si cifre, ori numere care nu apartin domeniului specific meniului ce cuprinde functionalitatile la care are acces userul in acel moment, se va afisa un mesaj corespunzator.

User-ul nu poate introduce o valoarea **negativa** a sumei de bani – in cazul in care se doreste depozitare, se va transmite valoarea nemodificata, insa, in cazul in care se doreste extragerea, de va transmite functiei de depozitare opusul valorii introduse de user (adica valoarea negativa).

Ori de cate ori apare un nou client, se **realoca** memorie (**dinamic**)

Dupa citirea unui numar (fie ca e intreg sau real), trebuie **extras** din **buffer** caracterul **new line**: ‘\n’ inaintea unei citiri de sir de caractere

In cazul **exportarii** datelor financiare, user-ul trebuie sa introduca **numele fisierului** in care vor fi stocate informatiile si, in functie de preferinta sa, se va atasa, la finalul denumirii fisierului, extensia **‘.csv’** ori **‘.txt’**

La sfarsitul programului, trebuie **eliberata** memoria alocata **dinamic** (atat pentru **clientii** tuturor user-ilor, cat si pentru toti **userii** insisi, cu ajutorul functiei ‘**free’**).