**Aplicaţie**

Un investitor deţine un fond de 30000 Euro şi are de ales între 2 variante de investiţie pentru o perioadă de 1 lună:

1. Să consituie un depozit la bancă cu dobânda de 5% pe an;
2. Să aloce banii pentru acordarea de împrumuturi pe termen scurt în cazul unei case de amanet. El dispune de următoarele informaţii privind activitatea casei de amanet :

-numărul clienţilor ce solicită în fiecare zi împrumut este o variabilă aleatoare discrete cu urmatoarea lege:

N~

-cererea unui client este tot o variabilă aleatoare discretă cu legea:

C~

-numărul de zile pentru care se solicită împrumut este o tot variabilă aleatoare cu legea:

Nz~

-dobânda zilnică percepută de casa de amanet în funcţie de volumul cererii şi de durată este ilustrată în tabelul:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 zile | 3 zile | 4 zile |
| 1000€ | 0.1% | 0.5% | 1% |
| 2000€ | 0.5% | 1% | 1.5% |
| 3000€ | 1% | 1.5% | 2% |

Ipotezele in care isi desfasoara activitatea casa de amanet sunt urmatoarele:

* 1. Casa de amanet funcţionează non-stop;
  2. Dobânda se plăteşte la acordarea împrumutului;
  3. Imprumutul nu se poate anula şi decurge din ziua efectuării împrumutului;
  4. Dacă cererea zilnică > disponibilul zilnic, atunci nu se acordă împrumutul;
  5. Cheltuielile de funcţionare ale casei de amanet: 50€/zi.

Se cere:

Sa se verifice prin metoda simularii numerice daca investitia in casa de amanet este mai profitabila decat un depozit constituit la banca. Realizati un program prin care veti calcula doua cicluri de simulare.

**Rezolvare:**

***1 Pregătirea simulării:***

Orizontul de simulare este de 30 de zile.

Exista 3 variabile aleatoare:

|  |  |  |  |
| --- | --- | --- | --- |
| N | *pi* |  *pi* | Interval |
| 0 | 0.2 | 0.2 | [0, 0.2) |
| 1 | 0.2 | 0.4 | [0.2,0.4) |
| 2 | 0.5 | 0.9 | [0.4,0.9) |
| 3 | 0.1 | 1 | [0.9,1] |

|  |  |  |  |
| --- | --- | --- | --- |
| C | *pi* |  *pi* | Interval |
| 1000€ | 0.5 | 0.5 | [0, 0.5) |
| 2000€ | 0.3 | 0.8 | [0.5,0.8) |
| 3000€ | 0.2 | 1 | [0,8,1] |

|  |  |  |  |
| --- | --- | --- | --- |
| *Nz* | *pi* |  *pi* | Interval |
| 2 zile | 0.3 | 0.3 | [0, 0.3) |
| 3 zile | 0.6 | 0.9 | [0,3, 0.9) |
| 4 zile | 1 | 1 | [0.9, 1] |

***2 Simularea propriu-zisă a activitatii casei de amanet***

Am ales limbajul C++ pentru a rezolva problema, pe care am impartit-o in doua: functiile si corpul principal.  
Observatie: // reprezinta comentariu, ignorat de compilator

1. **Functiile**

**a)** **Functia pentru generarea numerelor cuprinse in intervalul [0,1]**:

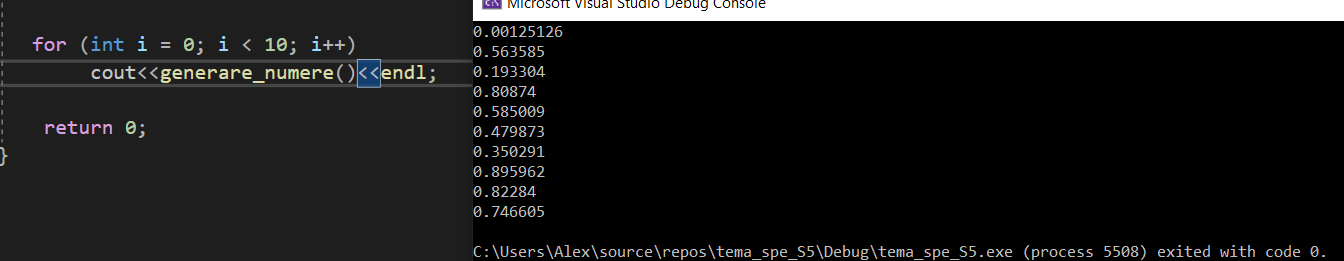
float generare\_numere()

{

return (float)rand() / RAND\_MAX;

}

Exemplu de apel al functiei pentru 10 elemente:



**b)** **Functia care decide cati clienti vor fi in acea zi la casa de amanet**. Aceasta functie se bazeaza pe functia anterioara: primeste ca parametru un numar generat de functia de la subpunctul a):

int decizie\_nr\_clienti(float numar\_generat)

{

if (numar\_generat < 0.2)

return 0;

else

if (numar\_generat >= 0.2 && numar\_generat < 0.4)

return 1;

else

if (numar\_generat >= 0.4 && numar\_generat < 0.9)

return 2;

else

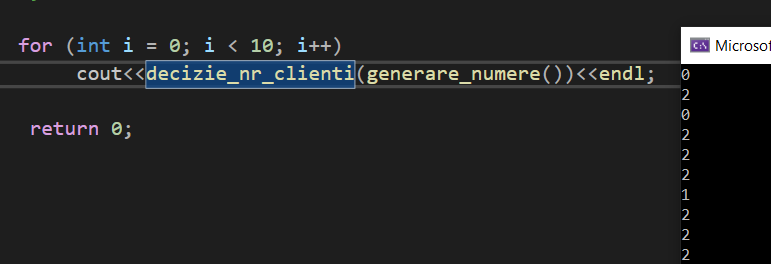
if (numar\_generat >= 0.9 && numar\_generat <= 1)

return 3;

}

Aceasta functie returneaza ca rezultate posibile 0, 1, 2 sau 3 clienti, in functie de intervalul caruia apartine numarul generat.

Exemplu de apel al functiei, pentru 10 elemente:



**c)** **Functia care decide ce valoare are creditul luat de client**. Asemenea functiei anterioare, primeste ca parametru un numar generat intre [0,1] si rezultatul returnat depinde de intervalul caruia apartine parametrul.

int decizie\_val\_credit(float numar\_generat)

{

if (numar\_generat < 0.5)

return 1000;

else

if (numar\_generat >= 0.5 && numar\_generat < 0.8)

return 2000;

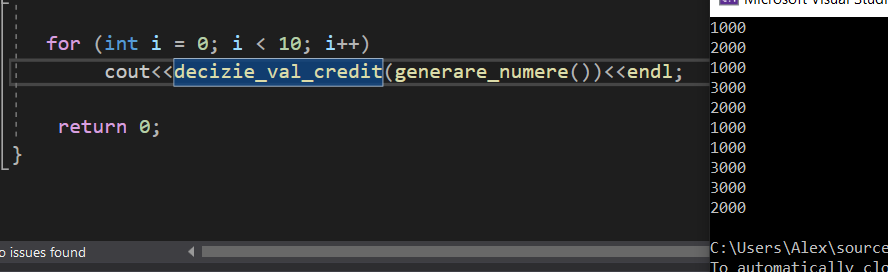
else

if (numar\_generat >= 0.8 && numar\_generat <= 1)

return 3000;

}

Exemplu de apel pentru 10 elemente:



**d)** **Functia care decide pentru cate zile va fi acordat creditul**. Primeste ca parametru un numar generat de functia generare\_numar.

int decizie\_nr\_zile(float numar\_generat)

{

if (numar\_generat < 0.3)

return 2;

else

if (numar\_generat >= 0.3 && numar\_generat < 0.9)

return 3;

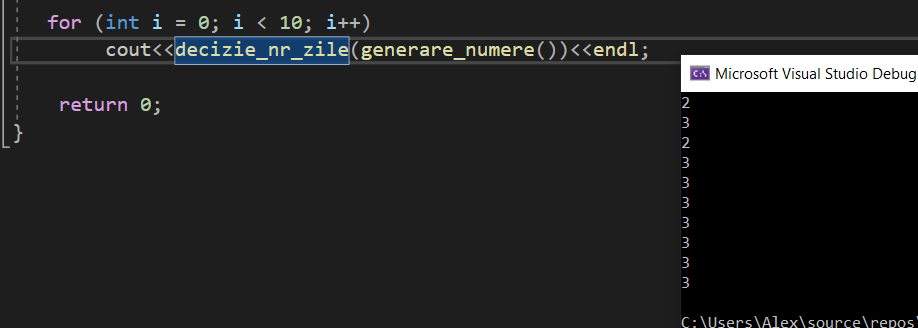
else

if (numar\_generat >= 0.9 && numar\_generat <=1)

return 4;

}

Exemplu de apel pentru 10 elemente:



**e) Functia care determina perechile (valoare\_credit, nr\_zile) si numarul de clienti.** Aceasta functie foloseste toate celelalte functii anterior mentionate pentru a determina numarul de clienti al unei zile, rezultat pe care il returneaza. De asemenea, functia primeste ca parametrii doi vectori (cu spatiu pentru maxim 3 elemente-numarul maxim de clienti al unei zile), in care salveaza valoarea creditului si numarul de zile pentru fiecare client i, pe pozitia i a fiecarui vector, unde i ia valori de la 0 la numarul clientilor mai sus generat.

int vector\_clienti(int credit[3], int zile[3])

{

int nr\_clienti;

nr\_clienti = decizie\_nr\_clienti(generare\_numere()); // decide cati clienti vor fi in ziua respectiva,

for (int i = 0; i < nr\_clienti; i++) //pt fiecare client genereaza valoarea creditului si numarul de zile

{

credit[i] = decizie\_val\_credit(generare\_numere());// genereaza valoarea creditului

zile[i] = decizie\_nr\_zile(generare\_numere());// genereaza numarul de zile

}

return nr\_clienti; // returneaza numarul de clienti

}

**f) Functia care calculeaza dobanda**. Functia primeste doi parametri, valoarea creditului si numarul de zile, pe baza carora calculeaza dobanda.

float dobanda(int valoare\_credit, int nr\_zile)

{

if (valoare\_credit == 1000)

{

if (nr\_zile == 2)

return 0.001\*valoare\_credit\*nr\_zile;

else

if (nr\_zile == 3)

return 0.005 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.01 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

if (valoare\_credit == 2000)

{

if (nr\_zile == 2)

return 0.005 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 3)

return 0.01 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.015 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

if (valoare\_credit == 3000)

{

if (nr\_zile == 2)

return 0.01 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 3)

return 0.015 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.02 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

cout << endl << endl << "VALOARE GRESITA A CREDITULUI " << endl;

}

1. **Corpul principal al codului**

Am ales situatia in care casa de amanet ofera credite timp de 26 de zile, ca la sfarsitul orizontului de simulare de 30 de zile, toate creditele sa fie restituite. Astfel, daca un client cere un credit de 4 zile in ziua a 26-a, acesta il va returna in ziua a 30-a.

* Voi declara pentru fiecare dintre primele 5 zile variabile unice, in care voi stoca informatiile despre numarul de clienti, valoarea creditelor si numarul de zile pentru care este acordat. Pentru fiecare zi voi calcula dobanda provenita de la clienti, (in variabila dobanda\_i), soldul disponibil, profitul zilnic (profit\_zilnic) si voi actualiza suma profitelor zilnicem in variabila profit.
* Cea mai mare valoarea a creditului este de 3000 Euro, iar numarul maxim posibil de clienti intr-o zi este de 3. Astfel, in primele 3 zile, se vor da credite in valoare maxima de 3x3x3000=27000 Euro, ceea ce este mai mic decat soldul initial, de 30000 Euro. Problema verificarii situatiei in care soldul disponibil nu poate sustine toare cererile se pune abia din ziua a 4a, cel mai devreme. Structurile repetitive verifica daca suma creditelor din ziua respectiva impreuna cu cheltuielile de intretinere depasesc soldul disponibil, si in caz afirmativ, nu mai acorda creditul ultimului client.
* Din ziua a 5a se pot reutiliza variabilele, deoarece daca in prima zi am avut credite acordate pe 2 zile, acestea s-au predat in ziua 3, creditele pe 3 zile au fost predate in ziua 4, iar creditele pe 4 zile au fost predate in ziua 5. Astfel, in variabilele alocate zilei 1 nu mai exista informatii necesare. Algoritmul poate fi generalizat, mutand toate informatiile cu o zi in urma, si creand bucle repetitive.
* La final se verifica daca profitul mediu din 21 cicluri adus de investitia in casa de amanet este mai mare decat 125 Euro, dobanda primita de la banca in situatia depunerii sumei de 30000Euro, si se afiseaza un mesaj corespunzator.

***Corpul principal:***

int main()

{

float profit\_mediu = 0;

for (int j = 0; j < 2; j++)

{

int profit = 0;

//ziua 1

int c1[3], z1[3], clienti1, dobanda\_i = 0, sold = 30000, profit\_zilnic; // dobanda\_i reprezinta dobanda pe o zi

clienti1 = vector\_clienti(c1, z1);

for (int i = 0; i < clienti1; i++)

{

dobanda\_i = dobanda\_i + dobanda(c1[i], z1[i]);

sold = sold - c1[i];

}

sold = sold - 50 + dobanda\_i;

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 2

int c2[3], z2[3], clienti2;

dobanda\_i = 0;

clienti2 = vector\_clienti(c2, z2);

for (int i = 0; i < clienti2; i++)

{

dobanda\_i = dobanda\_i + dobanda(c2[i], z2[i]);

sold = sold - c2[i];

}

sold = sold - 50 + dobanda\_i;

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 3

int c3[3], z3[3], clienti3;

dobanda\_i = 0;

clienti3 = vector\_clienti(c3, z3);

for (int i = 0; i < clienti3; i++)

{

dobanda\_i = dobanda\_i + dobanda(c3[i], z3[i]);

sold = sold - c3[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti1; i++)

if (z1[i] == 2)

sold = sold + c1[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

// ziua 4

int sum = 0;

int c4[3], z4[3], clienti4;

dobanda\_i = 0;

clienti4 = vector\_clienti(c4, z4);

for (int i = 0; i < clienti4; i++)

sum = c4[i] + sum;

while ((sum + 50 > sold) && (clienti4 != 0)) // se verifica daca soldul disponibil poate suporta toate creditele clientilor generati in ziua respectiva; in cazul in care suma creditelor din ziua curenta +cheltuielile de intretinere este mai mare decat soldul disponibil numarul clientilor se scade si se reface suma pentru o noua verificare, care se termina atunci cand suma nu depaseste soldul disponibil sau daca numarul clientilor din ziua curenta a ajuns sa fie 0

{

clienti4--;

sum = 0;

for (int i = 0; i < clienti4; i++)

sum = c4[i] + sum;

}

for (int i = 0; i < clienti4; i++)

{

dobanda\_i = dobanda\_i + dobanda(c4[i], z4[i]);

sold = sold - c4[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti2; i++)

if (z2[i] == 2)

sold = sold + c2[i];

for (int i = 0; i < clienti1; i++)

if (z1[i] == 3)

sold = sold + c1[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 5+

for (int i = 5; i < 27; i++)

{

sum = 0;

int c5[3], z5[3], clienti5;

dobanda\_i = 0;

clienti5 = vector\_clienti(c5, z5);

for (int i = 0; i < clienti5; i++)

sum = c5[i] + sum;

while ((sum + 50 > sold) && (clienti5 != 0))

{

clienti5--;

sum = 0;

for (int i = 0; i < clienti5; i++)

sum = c5[i] + sum;

}

for (int i = 0; i < clienti5; i++)

{

dobanda\_i = dobanda\_i + dobanda(c5[i], z5[i]);

sold = sold - c5[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti1; i++) // verific daca am credite de acum 4 zile care trebuie sa fie predate azi

if (z1[i] == 4)

sold = sold + c1[i];

for (int i = 0; i < clienti2; i++) // verific daca am credite de acum 3 zile care trebuie sa fie predate azi

if (z2[i] == 3)

sold = sold + c2[i];

for (int i = 0; i < clienti3; i++) // verific daca am credite de acum 2 zile care trebuie sa fie predate azi

if (z3[i] == 2)

sold = sold + c3[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

for (int i = 0; i < clienti2; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c1[i] = c2[i]; z1[i] = z2[i];

}

clienti1 = clienti2;

for (int i = 0; i < clienti3; i++)//mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c2[i] = c3[i]; z2[i] = z3[i];

}

clienti2 = clienti3;

for (int i = 0; i < clienti4; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c3[i] = c4[i]; z3[i] = z4[i];

}

clienti3 = clienti4;

for (int i = 0; i < clienti5; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c4[i] = c5[i]; z4[i] = z5[i];

}

clienti4 = clienti5;

profit = profit + profit\_zilnic;

}

cout << endl << endl << endl << "Profitul total =suma din profitul zilnic: " << profit << endl

<< "Profitul mediu zilnic = profitul total impartit la cate profituri zilnice am (26): " << profit / 26;

cout << endl << endl << endl;

profit\_mediu = profit\_mediu + profit / 26;

}

cout << profit\_mediu / 2 << "=profitul mediu pt 2 cicluri"<<endl;

if (profit\_mediu / 2 < 125)

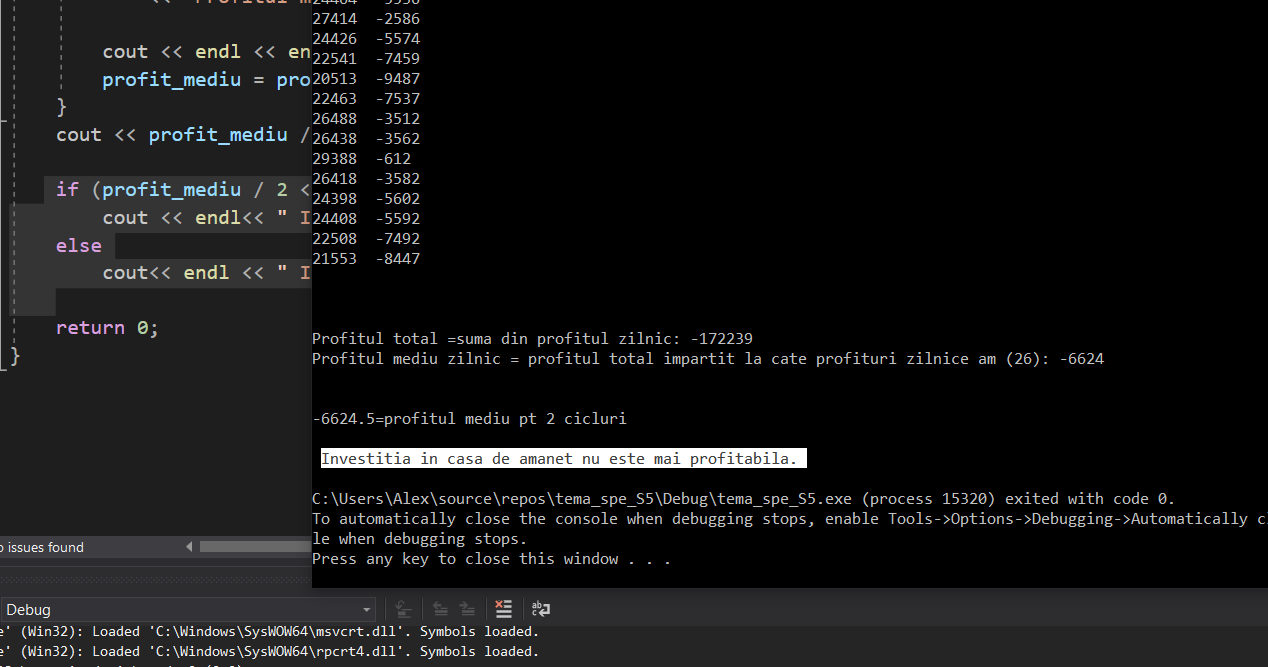
cout << endl<< " Investitia in casa de amanet nu este mai profitabila." << endl;

else

cout<< endl << " Investitia in casa de amanet este mai profitabila." << endl;

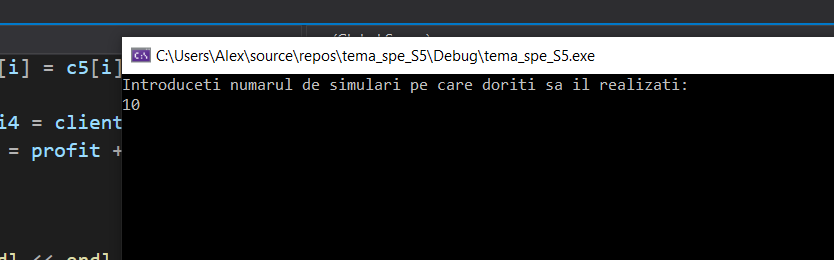
return 0;

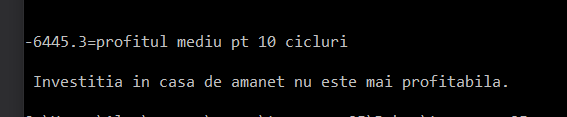
}



In urma a 2 cicluri de simulare, conform algoritmului utilizat, investitia in casa de amanet este nu este mai profitabila decat depozitul la banca.

Codul poate fi usor modificat pentru a calcula pentru n cicluri de simulare, unde n este introdus de utilizator.





**Anexa**

**Codul integral utilizat la rezolvarea problemei (cel initial, cu 2 cicluri de simulare):**

#include <iostream>

#include <stdlib.h>

using namespace std;

float generare\_numere()

{

return (float)rand() / RAND\_MAX;

}

int decizie\_nr\_clienti(float numar\_generat)

{

if (numar\_generat < 0.2)

return 0;

else

if (numar\_generat >= 0.2 && numar\_generat < 0.4)

return 1;

else

if (numar\_generat >= 0.4 && numar\_generat < 0.9)

return 2;

else

if (numar\_generat >= 0.9 && numar\_generat <= 1)

return 3;

}

int decizie\_val\_credit(float numar\_generat)

{

if (numar\_generat < 0.5)

return 1000;

else

if (numar\_generat >= 0.5 && numar\_generat < 0.8)

return 2000;

else

if (numar\_generat >= 0.8 && numar\_generat <= 1)

return 3000;

}

int decizie\_nr\_zile(float numar\_generat)

{

if (numar\_generat < 0.3)

return 2;

else

if (numar\_generat >= 0.3 && numar\_generat < 0.9)

return 3;

else

if (numar\_generat >= 0.9 && numar\_generat <=1)

return 4;

}

int vector\_clienti(int credit[3], int zile[3])

{

int nr\_clienti;

nr\_clienti = decizie\_nr\_clienti(generare\_numere()); // decide cati clienti vor fi in ziua respectiva,

for (int i = 0; i < nr\_clienti; i++) //pt fiecare client genereaza valoarea creditului si numarul de zile

{

credit[i] = decizie\_val\_credit(generare\_numere());

zile[i] = decizie\_nr\_zile(generare\_numere());

}

return nr\_clienti; // returneaza numarul de clienti sa nu lucrezi in partea de vector null

}

float dobanda(int valoare\_credit, int nr\_zile) // functie care calculeaza valoarea dobanzii;

{

if (valoare\_credit == 1000)

{

if (nr\_zile == 2)

return 0.001\*valoare\_credit\*nr\_zile;

else

if (nr\_zile == 3)

return 0.005 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.01 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

if (valoare\_credit == 2000)

{

if (nr\_zile == 2)

return 0.005 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 3)

return 0.01 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.015 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

if (valoare\_credit == 3000)

{

if (nr\_zile == 2)

return 0.01 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 3)

return 0.015 \* valoare\_credit \* nr\_zile;

else

if (nr\_zile == 4)

return 0.02 \* valoare\_credit \* nr\_zile;

else

cout << endl << endl << "VALOARE GRESITA A NUMARULUI DE ZILE" << endl;

}

else

cout << endl << endl << "VALOARE GRESITA A CREDITULUI " << endl;

}

int main()

{

float profit\_mediu = 0;

for (int j = 0; j < 2; j++)

{

int profit = 0;

//ziua 1

int c1[3], z1[3], clienti1, dobanda\_i = 0, sold = 30000, profit\_zilnic;

clienti1 = vector\_clienti(c1, z1);

for (int i = 0; i < clienti1; i++)

{

dobanda\_i = dobanda\_i + dobanda(c1[i], z1[i]);

sold = sold - c1[i];

}

sold = sold - 50 + dobanda\_i;

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 2

int c2[3], z2[3], clienti2;

dobanda\_i = 0;

clienti2 = vector\_clienti(c2, z2);

for (int i = 0; i < clienti2; i++)

{

dobanda\_i = dobanda\_i + dobanda(c2[i], z2[i]);

sold = sold - c2[i];

}

sold = sold - 50 + dobanda\_i;

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 3

int c3[3], z3[3], clienti3;

dobanda\_i = 0;

clienti3 = vector\_clienti(c3, z3);

for (int i = 0; i < clienti3; i++)

{

dobanda\_i = dobanda\_i + dobanda(c3[i], z3[i]);

sold = sold - c3[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti1; i++)

if (z1[i] == 2)

sold = sold + c1[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

// ziua 4

int sum = 0;

int c4[3], z4[3], clienti4;

dobanda\_i = 0;

clienti4 = vector\_clienti(c4, z4);

for (int i = 0; i < clienti4; i++)

sum = c4[i] + sum;

while ((sum + 50 > sold) && (clienti4 != 0)) {

clienti4--;

sum = 0;

for (int i = 0; i < clienti4; i++)

sum = c4[i] + sum;

}

for (int i = 0; i < clienti4; i++)

{

dobanda\_i = dobanda\_i + dobanda(c4[i], z4[i]);

sold = sold - c4[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti2; i++)

if (z2[i] == 2)

sold = sold + c2[i];

for (int i = 0; i < clienti1; i++)

if (z1[i] == 3)

sold = sold + c1[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

profit = profit + profit\_zilnic;

//ziua 5+

for (int i = 5; i < 27; i++)

{

sum = 0;

int c5[3], z5[3], clienti5;

dobanda\_i = 0;

clienti5 = vector\_clienti(c5, z5);

for (int i = 0; i < clienti5; i++)

sum = c5[i] + sum;

while ((sum + 50 > sold) && (clienti5 != 0)) {

clienti5--;

sum = 0;

for (int i = 0; i < clienti5; i++)

sum = c5[i] + sum;

}

for (int i = 0; i < clienti5; i++)

{

dobanda\_i = dobanda\_i + dobanda(c5[i], z5[i]);

sold = sold - c5[i];

}

sold = sold - 50 + dobanda\_i;

for (int i = 0; i < clienti1; i++) // verific daca am credite de acum 4 zile care trebuie sa fie predate azi

if (z1[i] == 4)

sold = sold + c1[i];

for (int i = 0; i < clienti2; i++) // verific daca am credite de acum 3 zile care trebuie sa fie predate azi

if (z2[i] == 3)

sold = sold + c2[i];

for (int i = 0; i < clienti3; i++) // verific daca am credite de acum 2 zile care trebuie sa fie predate azi

if (z3[i] == 2)

sold = sold + c3[i];

profit\_zilnic = sold - 30000;

cout << sold << " " << profit\_zilnic << endl;

for (int i = 0; i < clienti2; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c1[i] = c2[i]; z1[i] = z2[i];

}

clienti1 = clienti2;

for (int i = 0; i < clienti3; i++)

{

c2[i] = c3[i]; z2[i] = z3[i];

}

clienti2 = clienti3;

for (int i = 0; i < clienti4; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c3[i] = c4[i]; z3[i] = z4[i];

}

clienti3 = clienti4;

for (int i = 0; i < clienti5; i++) //mut elementele din vectori sa fie la distanta egala fata de ziua actuala

{

c4[i] = c5[i]; z4[i] = z5[i];

}

clienti4 = clienti5;

profit = profit + profit\_zilnic;

}

cout << endl << endl << endl << "Profitul total =suma din profitul zilnic: " << profit << endl

<< "Profitul mediu zilnic = profitul total impartit la cate profituri zilnice am (26): " << profit / 26;

cout << endl << endl << endl;

profit\_mediu = profit\_mediu + profit / 26;

}

cout << profit\_mediu / 2 << "=profitul mediu pt 2 cicluri"<<endl;

if (profit\_mediu / 2 < 125)

cout << endl<< " Investitia in casa de amanet nu este mai profitabila." << endl;

else

cout<< endl << " Investitia in casa de amanet este mai profitabila." << endl;

return 0;

}