Tablouri bidimensionale(matrici)

varianta în C++

Problema 1: Secvenţă de matrici

Enunţ

Scrieţi un program care citeşte de la tastatură două numere naturale *n* şi *m* (*2<n<20*, *2<m<10000*) şi construieşte în memorie o secvenţă de matrici pătratice (*n* linii şi *n* coloane), pe care o afişează pe ecran.

Prima matrice (iniţială) va avea elementele de pe diagonala secundară egale cu *m*, după care fiecare element aflat deasupra diagonalei secundare va fi mai mic cu o unitate decât vecinul aflat pe aceeaşi linie în dreapta lui şi fiecare element aflat sub diagonala secundară va fi mai mare cu o unitate decât vecinul aflat pe aceeaşi linie în stânga lui.

Următoarele matrici (secundare) din serie vor conţine pe fiecare poziţie *(i, j)* numărul divizorilor proprii ai elementului *(i, j)* a matricii anterioare din secvenţă. Seria continuă cu matrici obţinute repetând acest mod de calcul până când vom avea o matrice cu toate elementele nule.

Exemplu

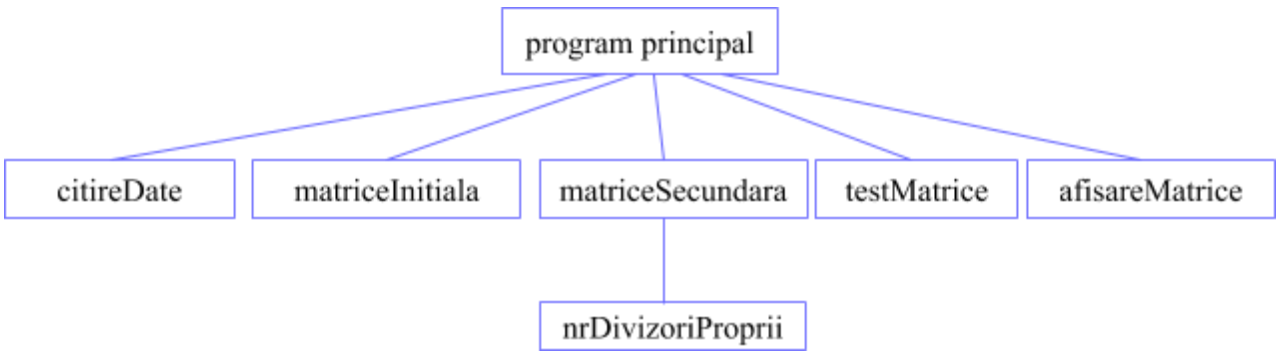
Date de intrare	Rezultate
n=5 m=47	43 44 45 46 47 44 45 46 47 48 45 46 47 48 49 46 47 48 49 50 47 48 49 50 51 0 4 4 2 0 4 4 2 0 8 4 2 0 8 1 2 0 8 1 4 0 8 1 4 2 0 1 1 1 0 1 1 1 0 2 1 1 0 2 0 1 0 2 0 1 0 2 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0

	0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Paşii algorimului principal

```
Algoritm secventaMatrici
     @ citeşte n şi m
     @ iniţializează prima matrice din secvenţă (după prima regulă)
     @ afişează prima matrice din secvenţă
     @ CâtTimp matricea nu este nulă execută
          @ generează noua matrice (după a doua regulă)
          @ afişează matricea curentă
     @ Sf.CâtTimp
Sf.Algoritm
```

Identificarea subalgoritmilor



Programul

Observaţii:

Nu s-a cerut ca indexarea să se facă de la 0 sau de la 1
Rezolvarea este data pentru matrici indexate de la unu
Sunt date două variante de construire a matricii iniţiale (una care urmăreşte direcţia diagonalei secundare şi una unde s-a folosit modul particular cum se aranjează numerele în acest tip de matrice)

```
#include <iostream>

using namespace std;

void citireDate(int & inputN,int & inputM)
{
```

```
  cout<<" Introduceti dimensiunea matricii n=";cin>>inputN;
  cout<<" Introduceti numarul m="; cin>>inputM;
}

void matriceInitiala1(int matrice[][21],int dimensiune,int m)
{
      // genereaza prima matrice din secventa conform primei reguli
      // se parcurge efectiv matricea pe directii paralele cu diagonala
      // secundara

      int i,j;

      for(i=1;i<=dimensiune;i++)
            matrice[i][dimensiune-i+1]=m;
      for (i=1;i<=dimensiune;i++)
            for (j=dimensiune-i;j>=1;j--)
                  matrice[i][j]=matrice[i][j+1]-1;
      for (i=2;i<=dimensiune;i++)
            for (j=dimensiune-i+2;j<=dimensiune;j++)
                  matrice[i][j]=matrice[i][j-1]+1;
}

void matriceInitiala2(int matrice[][21], int dimensiune, int m)
{
      // genereaza prima matrice din secventa conform primei reguli
      // observand modul cum ar arata valorile in ea

      int  i,j;

      for (j=dimensiune;j>=1;j--)
            matrice[1][j]=m-dimensiune+j;
      for (i=2;i<=dimensiune;i++)
            for (j=1;j<=dimensiune;j++)
                  matrice[i][j]=matrice[i-1][j]+1;
}

int nrDivizoriProprii(int nr)
{
      // numaram divizorii proprii ai argumentului functiei

      int i, numar;

      numar=0;
      if (nr>2)
      {
      for (i=2;i<=(nr / 2)+1;i++)
            if ((nr % i)==0)
                  numar++;
      }
      return numar;
}

void matriceSecundara(int matrice[][21],int dimensiune)
```

```
{
      // generarea matricilor dupa a doua regula

      int i,j;
      for (i=1;i<=dimensiune;i++)
            for (j=1;j<=dimensiune;j++)
                  matrice[i][j]=nrDivizoriProprii(matrice[i][j]);
}

bool testMatrice(int matrice[][21],int dimensiune)
{
      // testam daca matricea este matricea nula

      bool indicator=true;
      int  i,j;

      for (i=1;i<=dimensiune;i++)
            for (j=1;j<=dimensiune;j++)
                  if (matrice[i][j]!=0)
                        indicator=false;
      return indicator;
}


void afisareMatrice(int matrice[][21],int dimensiune)
{
      int i,j;

      for (i=1;i<=dimensiune;i++)
      {
      cout<<endl;
      for (j=1;j<=dimensiune;j++)
            cout<<' '<< matrice[i][j];
      }
      cout<<endl;
}


int main()
{
      int n,m;
      citireDate(n,m);
      int a[21][21];

      matriceInitiala1(a,n,m); //matriceInitiala2(a,n,m);

      afisareMatrice(a,n);
      while (! testMatrice(a,n))
      {
            matriceSecundara(a,n);
            afisareMatrice(a,n);
      }
      return 0;
```

}

Problema 2: Sistem de ecuaţii liniare

Enunţ

Fie un sistem de *n* ecuaţii liniare cu *n* necunoscute ($2 \le n \le 10$). De exemplu pentru $n = 4$ sistemul arată astfel:

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 = b_1$$
$$a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + a_{2,4}x_4 = b_2$$
$$a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 + a_{3,4}x_4 = b_3$$
$$a_{4,1}x_1 + a_{4,2}x_2 + a_{4,3}x_3 + a_{4,4}x_4 = b_4$$

Rezolvaţi acest sistem folosind matrici şi/sau determinanţi. Coeficienţii sistemului precum şi termenii liberi sunt numere reale ce vor fi citite de la tastatură.

Exemplu

Date de intrare	Rezultate
n=3 matricea coeficientilor (3 x 3): 2 3 -1 -1 1 2 7 -1 -3 matricea termenilor liberi (3 x 1): 5 10 15	x[1]=5.40 x[2]=0.54 x[3]=7.43

Indicaţii: folosim metoda lui Cramer

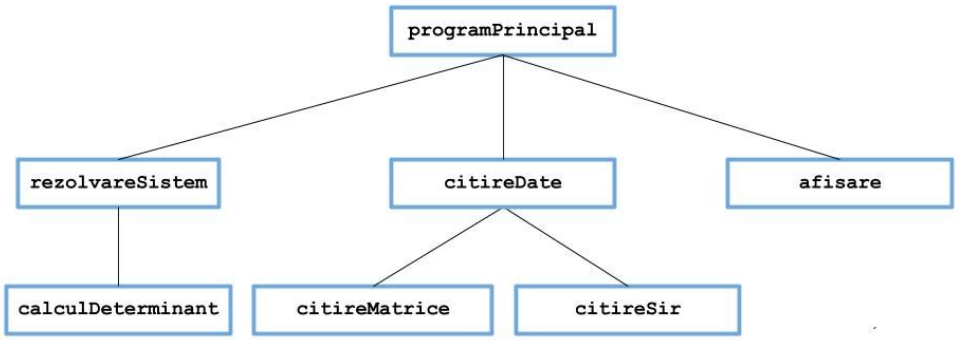
Paşii algorimului principal

Algoritm sistemEcuatii
- @ se calculează determinantul sistemului Δ
- @ dacă $\Delta \neq 0$ atunci
  - @ se calculează $x_i = \frac{\Delta x_i}{\Delta}$ unde $i = \overline{1,n}$, Δx_i fiind determinantul obţinut din Δ prin înlocuirea coloanei corespunzătoare coeficienţilor necunoscutei x_i, $i = \overline{1,n}$ cu coloana termenilor liberi
  - @ se afişează soluţia

@ altfel se afişează un mesaj că sistemul e incompatibil sau nedeterminat
Sf.Algoritm

Observăm că mai sunt şi alte metode de a rezolva sistemele liniare de ecuaţii folosind matrici însă aici vom exemplifica doar pe aceasta.

Identificarea subalgoritmilor



Programul

```
#include<iostream>

using namespace std;

float calculDeterminant(float d[][11],int dimensiune)
{
      // calculam recursiv un determinant prin dezvoltarea sa pe o line
      // daca dimensiunea lui e mai mare decat unu (alegem prima linie)
      // daca dimensiunea e unu valoarea sa e acel element unic

      int k,i,j, coef;
      float  aux[11][11], delta;

      delta=0; coef=1;
      if (dimensiune==1)
            delta=d[1][1];
      else
            {
            for (k=1;k<=dimensiune;k++)
                  {
                  for (i=2;i<=dimensiune;i++)
                        for (j=1;j<=dimensiune;j++)
                              {
                              if (j<k)
                                    aux[i-1][j]=d[i][j];
                              if (j>k)
                                    aux[i-1][j-1]=d[i][j];
                              };
```

```
                delta=delta+coef*d[1][k]*calculDeterminant(aux,
dimensiune-1);
                coef=coef*(-1);
                };
          }
     return delta;

}

bool rezolvareSistem(float A[][11],int dimensiune,float B[], float X[])
{
     // daca se poate se calculeaza in X solutia sistemului si
     // returneaza true
     // daca functia returneaza false


     float  aux[11][11];
     float  delta;
     int  k,i,j;
     delta=calculDeterminant(A,dimensiune);
     if (delta != 0)
          {
          for (k=1;k<=dimensiune;k++)
                {
                for (i=1;i<=dimensiune;i++)
                      for (j=1;j<=dimensiune;j++)
                            if (k==j)
                                  aux[i][j]=B[i];
                            else
                                  aux[i][j]=A[i][j];
                X[k]=calculDeterminant(aux, dimensiune)/delta;
                };
          return true;
          }
     else
          return false;
}


void citireMatrice(float a[][11],int nrLinii,int nrColoane)
{
     int i,j;
     for (i= 1; i<= nrLinii;i++)
          for (j= 1;j<=nrColoane;j++)
                cin>>a[i][j];
}


void citireSir(float a[], int dimensiune)
{
     int i;
     for (i=1;i<=dimensiune;i++)
          cin>>a[i];
```

```
}

void afisare(float X[],int dimensiune, bool stare)
{

      int i;
      cout<<endl;
      if (stare)
            {
            for (i=1;i<=dimensiune;i++)
                  cout<<"x("<<i<<")="<<X[i]<<endl;
            }
      else
            cout<<"Sistem incompatibil sau nedeterminat!";
}

void citireDate(float a[][11],float b[],int & dimensiune)
{
      cout<<"Introduceti numarul necunoscutelor n=";cin>>dimensiune;
      cout<<"Introduceti coeficientii necunoscutelor (matricea ";
      cout<<"sistemului"<<dimensiune<<"x"<<dimensiune<<")"<<endl;
      citireMatrice(a,dimensiune,dimensiune);
      cout<<"Introduceti termenii liberi (matricea termenilor liberi";
      cout<<dimensiune<<"x1)"<<endl;
      citireSir(b,dimensiune);
}

int main()
{

      float A[11][11],X[11],B[11];
      int n;

      citireDate(A,B,n);
      afisare(X,n,rezolvareSistem(A,n,B,X));

      return 0;
}
```

Problema 3: Jocuri de minime

Enunţ

Se citeşte o matrice pătratică cu *n* linii şi *n* coloane ($3 \le n \le 100$), cu elemente numere naturale din intervalul $[0,1000]$.

a) Calculaţi şi afişaţi (în orice ordine) valoarile minime din fiecare dintre cele 4 zone în care este împarţită matricea de catre cele două diagonale (elementele de pe diagonale nu aparţin nici unei zone).

b) Interschimbați circular în sens trigonometric (invers acelor de ceas) cele 4 valori minime, de ori câte ori ar apărea valoarea înlocuită în zona de destinație. Astfel, valoarea minimă din zona N va ajunge pe pozițiile tuturor valorilor minime din V, cea din V în locul celor din S, cea din S în locul celor din E, iar cea din E în locul celor din N. Afișați matricea rezultată.

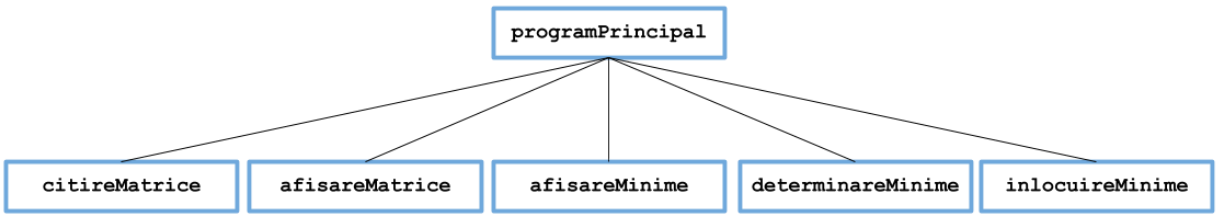
Exemplu

Date de intrare	Rezultate
n=4 2 3 10 8 2 1 2 1 2 11 5 7 0 4 8 3	 minim N=3 minim E=1 minim V=2 minim S=4 2 1 10 8 3 1 2 4 3 11 5 7 0 2 8 3

Pașii algorimului principal

```
Algoritm rotatiiMinime
      @ citește matricea
      @ determina valorile minime din cele patru zone
      @ afișează valorile minime
      @ rotește valorile minime intre zone
      @ afiseaza matricea noua
Sf.Algoritm
```

Identificarea subalgoritmilor



Programul

```
#include <iostream>

using namespace std;

void citireMatrice(int a[][101],int &dimensiune)
{
```

```
  int i,j;

  cout<<"Introduceti dimensiunea matricii n="; cin>>dimensiune;
  cout<<"Introduceti elementele matricii linie cu linie"<<endl;
  for (i=1;i<=dimensiune;i++)
     for (j=1;j<=dimensiune;j++)
          cin>>a[i][j];
}

void afisareMatrice(int a[][101],int dimensiune)
{
     int i,j;

     cout<<endl;
     for (i=1;i<=dimensiune;i++)
          {
          for (j=1;j<=dimensiune;j++)
               cout<<a[i][j]<<" ";
          cout<<endl;
          }
}


void afisareMinime(int s[])
{
  cout<<"minim N:"<<s[1]<<endl;
  cout<<"minim V:"<<s[2]<<endl;
  cout<<"minim S:"<<s[3]<<endl;
  cout<<"minim E:"<<s[4]<<endl;
}

void determinareMinime(int a[][101],int dimensiune, int sirMinime[])
{
   // determina cele patru valori minime N,V,S,E

   int i,j;
   for (i=1;i<= 4;i++)
      sirMinime[i]=1001;
   for (i=1;i<=dimensiune;i++)
      for (j=1;j<=dimensiune;j++)
          {
          if((i<j) && (i+j<dimensiune+1) && (a[i][j]<sirMinime[1]))
                    sirMinime[1]=a[i][j];
          if((i<j) && (i+j>dimensiune+1) && (a[i][j]<sirMinime[2]))
                    sirMinime[2]=a[i][j];
          if((i>j) && (i+j>dimensiune+1) && (a[i][j]<sirMinime[3]))
                    sirMinime[3]=a[i][j];
          if((i>j) && (i+j<dimensiune+1) && (a[i][j]<sirMinime[4]))
                    sirMinime[4]=a[i][j];
          };
}

void inlocuireMinime(int a[][101],int dimensiune, int sirMinime[])
```

```
{
   // inlocuieste valorile intre ele conform cerintei
   int  i,j;

   for (i=1;i<=dimensiune;i++)
      for (j=1;j<=dimensiune;j++)
           {
           if((i<j) && (i+j<dimensiune+1) && (a[i][j]==sirMinime[1]))
                      a[i][j]=sirMinime[2];
           if((i<j) && (i+j>dimensiune+1) && (a[i][j]==sirMinime[2]))
                      a[i][j]=sirMinime[3];
           if((i>j) && (i+j>dimensiune+1) && (a[i][j]==sirMinime[3]))
                      a[i][j]=sirMinime[4];
           if((i>j) && (i+j<dimensiune+1) && (a[i][j]==sirMinime[4]))
                      a[i][j]=sirMinime[1];
           }
}


int main()
{
      int matrice[101][101];
      int n;
      int minime[5];
      citireMatrice(matrice,n);
      determinareMinime(matrice,n,minime);
      afisareMinime(minime);
      inlocuireMinime(matrice,n,minime);
      afisareMatrice(matrice,n);

      return 0;
}
```

Problema 4: Jocul cu imagini

Enunţ

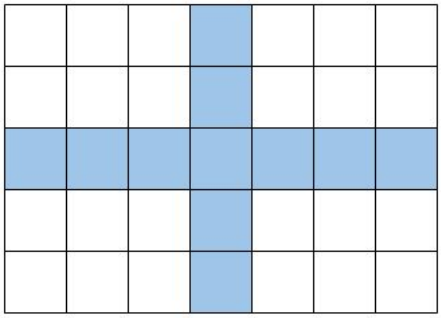
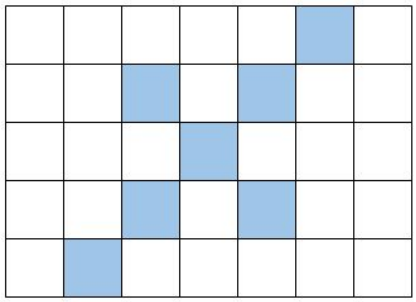
Un tânăr începe să se iniţieze în secretele prelucrării de imagini. Cu surprindere el află că o imagine alb negru poate fi reprezentată în principiu şi ca o matrice de zero şi unu şi că folosindu-se de înmulţirea matricilor el poate deforma imaginea. Cu toate acestea el nu ştie să înmulţească două matrici şi sa transforme o imagine. Ajutaţi-l să rotească o imagine cu un unghi θ.

Pentru asta scrieţi o funcţie ce calculează produsul a două matrici. Citiţi de la tastatură imaginea ca o matrice cu dimensiunea $n \times m$ ce va conţine doar zero şi unu precum şi matricea R ce o va folosi pentru rotire (de dimensiunea 2×2 ce conţine patru numere reale - coeficienţii transformării în vederea rotirii).

Pentru fiecare punct negru (1 în matrice) se calculează coordonatele inițiale (x_i, y_i), unde x_i e linia minus $n/2+n\%2$ și y_i este respectiv coloana minus $m/2+n\%2$ (pentru a stabili originea sistemului de coordonate în mijlocul matricii).

Noile coordonate (x_t, y_t) ale punctului după rotire se găsesc prin înmulțirea vectorului $[x_i, y_i]$ cu matricea transformărilor R. Acest punct va fi în matricea nouă la poziția (i, j) obținută din noile coordonate la care se adăugă valorile scăzute anterior când am stabilit originea sistemului de referință în mijlocul matricii. Dacă indicii noi pentru un punct sunt negativi sau depășesc dimensiunile *m x n* înseamnă că punctele respective ies din imaginea finală și nu mai sunt puse.

Exemplu

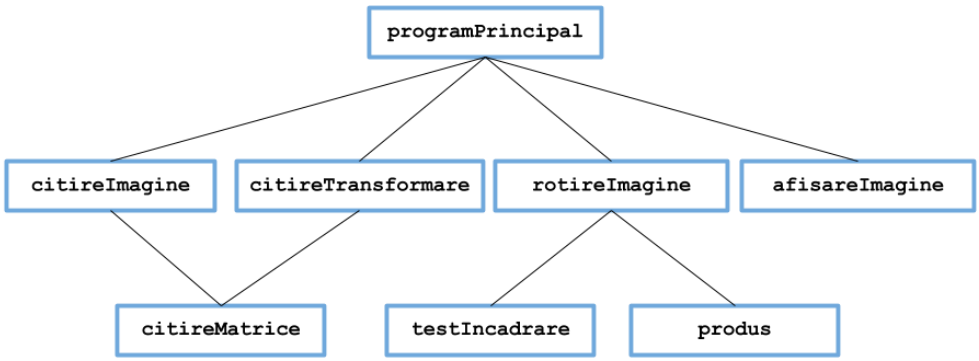
Date de intrare	Rezultate
Imaginea: n=5 m=7  0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 Rotirea (pt. un unghi de ~45 grade): 0.7 -0.7 0.7 0.7	Imaginea noua rotita:  0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0

Pașii algorimului principal

```
Algoritm joculCuImagini
     @ citește matricea imaginii
     @ citeste matricea rotatiei
     @ roteste imaginea
          @ pentru fiecare element al matricii imagine egal cu 1
               @ calculam coordonatele in raport cu centrul
               imaginii
```

```
                @ calculam noile coordonate dupa rotire
                @ calculam indicii i,j corespunzatori noilor
                coordonate in noua imagine
                @ punem valoarea 1 in matricea noua la pozitia i,j
      @ afisam matricea imaginii rotite
Sf.Algoritm
```

Identificarea subalgoritmilor



Programul

```
#include <iostream>

using namespace std;

struct matrix
{
      float a[100][100];
      int n,m;
};

void citireMatrice(matrix &mat)
{
      int i,j;
      for (i=1;i<=mat.n;i++)
      for (j=1;j<=mat.m;j++)
                cin>>mat.a[i][j];
}

void citireImagine(matrix &imagine)
{
   cout<<"Introduceti inaltimea imaginii n=";cin>>imagine.n;
   cout<<"Introduceti latimea imaginii m=";cin>>imagine.m;
   cout<<"Introduceti valoarea punctelor (0 sau 1) linie cu
line:"<<endl;
   citireMatrice(imagine);
}

void citireTransformare(matrix &rot)
{
   rot.n=2;
```

```
   rot.m=2;
    cout<<"Introduceti matricea transformarii cu dimensiunea 2 x 2 linie
cu linie"<<endl;
   citireMatrice(rot);
}


void afisareImagine(matrix imag)
{
      int i,j;

      for (i=1;i<=imag.n;i++)
      {
            cout<<endl;
            for (j=1;j<=imag.m;j++)
            cout<<(int)imag.a[i][j]<<' ';
      }

}

void produs(matrix m1, matrix m2, matrix &c)
{
   // calculeaza produsul matricilor m1 si m2. Rezultatul se pune in c

   int i,j,k;
   if (m1.m!=m2.n)
      {
            c.n=0;
            c.m=0;
      }
      else
      {
            c.n=m1.n;
            c.m=m2.m;
            for (i=1;i<=m1.n;i++)
                  for (j=1;j<=m2.m;j++)
                  {
                        c.a[i][j]=0;
                        for (k=1;k<=m1.m;k++)
                        c.a[i][j]=c.a[i][j]+m1.a[i][k]*m2.a[k][j];
                  }
      }
}

bool testIncadrare(int x, int y,int n,int m)
{
   //testeaza daca indicii x si y sunt in intevalele corecte

   return ((x>=1)and(x<=n)and(y>=1)and(y<=m));
}

void rotireImagine(matrix img, matrix R,matrix &newImg)
{
```

```
      //roteste imaginea conform coeficientilor din R
   matrix coordInitiale, coordTransf;

   int i,j, x,y;
   int ajustareX,ajustareY;

   newImg.n=img.n;
   newImg.m=img.m;
   coordInitiale.n=2;
   coordInitiale.m=1;

   ajustareX=img.n / 2+img.n % 2;
   ajustareY=img.m / 2+img.m % 2;

   for (i=1;i<=newImg.n;i++)
      for (j=1;j<=newImg.m;j++)
            newImg.a[i][j]=0;
   for (i=1;i<=img.n;i++)
      for (j=1;j<=img.m;j++)
            if (img.a[i][j]==1)
            {
                  coordInitiale.a[1][1]=i-ajustareX;
                  coordInitiale.a[2][1]=j-ajustareY;

                  produs(R, coordInitiale, coordTransf);

                 x=(int)coordTransf.a[1][1]+ajustareX;
                  y=(int)coordTransf.a[2][1]+ajustareY;

                  if (testIncadrare(x,y,newImg.n, newImg.m))
                        newImg.a[x][y]=1;
            }
}

int main()
{
      matrix img, rot, imgNew;
      citireImagine(img);
      citireTransformare(rot);
      rotireImagine(img,rot,imgNew);
      afisareImagine(imgNew);

      return 0;
}
```