Tablouri bidimensionale(matrici)

varianta în Pascal

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

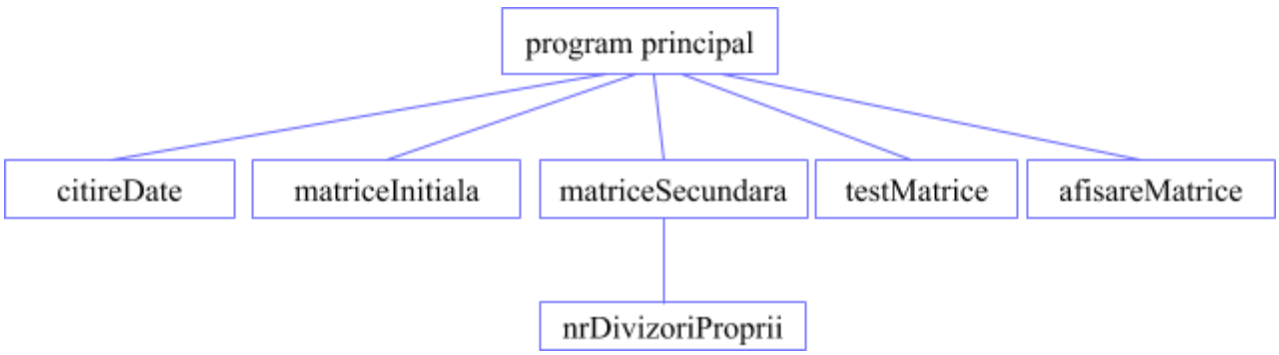
Date de intrare	Rezultate
n=5 m=47	43 44 45 46 47 44 45 46 47 48 45 46 47 48 49 46 47 48 49 50 47 48 49 50 51 0 4 4 2 0 4 4 2 0 8 4 2 0 8 1 2 0 8 1 4 0 8 1 4 2 0 1 1 1 0 1 1 1 0 2 1 1 0 2 0 1 0 2 0 1 0 2 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0

	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)

```
program secventaMatrici;

type matrix=array[1..20,1..20] of integer;

var
```

```
  n,m:integer;
  a:matrix;

procedure citireDate(var inputN:integer; var inputM:integer);

begin
  writeln();
  write(' Introduceti dimensiunea matricii n='); readln(inputN);
  write(' Introduceti numarul m='); readln(inputM);
end;

procedure matriceInitiala1(var matrice:matrix; dimensiune:integer;
m:integer);
// genereaza prima matrice din secventa conform primei reguli
// se parcurge efectiv matricea pe directii paralele cu diagonala
// secundara

var
  i,j:integer;
begin
  for i:=1 to dimensiune do
      matrice[i][dimensiune-i+1]:=m;
  for i:=1 to dimensiune do
      for j:=dimensiune-i downto 1 do
          matrice[i][j]:=matrice[i][j+1]-1;
  for i:=2 to dimensiune do
      for j:=dimensiune-i+2 to dimensiune do
          matrice[i][j]:=matrice[i][j-1]+1
end;

procedure matriceInitiala2(var matrice:matrix; dimensiune:integer;
m:integer);
// genereaza prima matrice din secventa conform primei reguli
// se genereaza matricea observand modul cum ar arata valorile in ea

var
  i,j:integer;
begin
  for j:=dimensiune downto 1 do
      matrice[1][j]:=m-dimensiune+j;
  for i:=2 to dimensiune do
      for j:=1 to dimensiune do
          matrice[i][j]:=matrice[i-1][j]+1;
end;

function nrDivizoriProprii(nr:integer):integer;
// numaram divizorii proprii ai argumentului functiei

var
  i, numar:integer;
begin
  numar:=0;
  if nr>2 then
```

```
      for i:=2 to (nr div 2)+1 do
         if (nr mod i)=0 then
              numar:=numar+1;
  nrDivizoriProprii:=numar;
end;

procedure matriceSecundara(var matrice:matrix; dimensiune:integer);
// generarea matricilor dupa a doua regula

var
  i,j:integer;
begin
  for i:=1 to dimensiune do
      for j:=1 to dimensiune do
          matrice[i][j]:=nrDivizoriProprii(matrice[i][j]);
end;

function testMatrice(matrice:matrix; dimensiune:integer):boolean;
// testam daca matricea este matricea nula

var
  indicator:boolean;
  i,j:integer;
begin
  indicator:=true;
  for i:=1 to dimensiune do
      for j:=1 to dimensiune do
          if matrice[i][j]<>0 then
                 indicator:=false;
  testMatrice:=indicator;
end;


procedure afisareMatrice(matrice:matrix; dimensiune:integer);
var
  i,j:integer;
begin
  for i:=1 to dimensiune do
      begin
        writeln();
        for j:=1 to dimensiune do
            write(' ', matrice[i][j]);
      end;
  writeln();
end;

begin
  citireDate(n,m);
  matriceInitiala1(a,n,m); //matriceInitiala2(a,n,m);
  afisareMatrice(a,n);
  while (not testMatrice(a,n)) do
     begin
        matriceSecundara(a,n);
```

```
        afisareMatrice(a,n);
      end;
  readln();
end.
```

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]=5.41 x[2]=0.54 x[3]=7.43

Indicaţii: folosim metoda lui Cramer

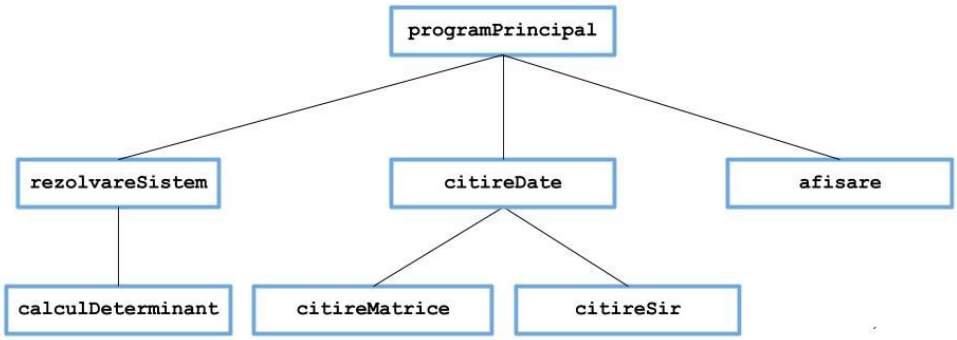
Paşii algorimului principal

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

```
program problema2;

type
  matrix = array [1..10,1..10] of real;
  sir = array [1..10] of real;

var
  A:matrix;
  X,B:sir;
  n:integer;

function calculDeterminant(d:matrix; dimensiune:integer):real;
// calculam recursiv un determinant prin dezvoltarea sa pe o line daca
// dimensiunea lui e mai mare decat unu (alegem arbitrar prima linie)
// daca dimensiunea e unu valoarea sa e acel element unic
var
  k,i,j, coef:integer;
  aux:matrix;
  delta:real;

begin
  delta:=0; coef:=1;
  if dimensiune=1 then
     delta:=d[1][1]
  else
    begin
      for k:=1 to dimensiune do
```

```
         begin
           for i:=2 to dimensiune do
               for j:=1 to dimensiune do
                   begin
                     if j<k then
                       aux[i-1][j]:=d[i][j];
                     if j>k then
                       aux[i-1][j-1]:=d[i][j];
                   end;
            delta:=delta+coef*d[1][k]*calculDeterminant(aux,
dimensiune-1);
            coef:=coef*(-1);
           end;

    end;
  calculDeterminant:=delta;
end;

function rezolvareSistem(A:matrix; dimensiune:integer; B:sir; var
X:sir):boolean;
// daca se poate se calculeaza in X solutia sistemului
// si returneaza true
// daca nu X ramane zero si functia returneaza false

var
  aux:matrix;
  delta:real;
  k,i,j:integer;
begin
  delta:=calculDeterminant(A,dimensiune);
  if delta <> 0 then
    begin
      rezolvareSistem:=true;
      for k:=1 to dimensiune do
          begin
            for i:=1 to dimensiune do
                for j:=1 to dimensiune do
                    if k=j then
                        aux[i][j]:=B[i]
                     else
                       aux[i][j]:=A[i][j];
            X[k]:=calculDeterminant(aux, dimensiune)/delta;
          end;
    end
  else
      rezolvareSistem:=false;
end;

procedure citireMatrice(var a:matrix; nrLinii:integer;
nrColoane:integer);

var
  i,j:integer;
```

```
begin
  for i:= 1 to nrLinii do
      begin
      for j:= 1 to nrColoane do
          read(a[i][j]);
      readln();
      end;
end;

procedure citireSir(var a:sir; dimensiune:integer);
var
  i:integer;
begin
  for i:=1 to dimensiune do
      read(a[i]);
  readln();
end;

procedure afisare(X:sir; dimensiune:integer; stare:boolean);
var
  i:integer;
begin
  writeln();
  if stare then
     begin
for i:=1 to dimensiune do
                           writeln('x(',i,')=',X[i]:2:2);
                end
           else
                writeln('Sistem incompatibil sau nedeterminat!');
end;


procedure citireDate(var a:matrix; var b:sir; var n:integer);

begin
  write('Introduceti numarul necunoscutelor n='); readln(n);
  writeln('Introduceti coeficientii necunoscutelor (matricea sistemului
',n,'x',n,')');
  citireMatrice(a,n,n);
  writeln('introduceti termenii liberi (matricea termenilor liberi
',n,'x1)');
  citireSir(b,n);
end;

begin
  citireDate(A,B,n);
  afisare(X,n, rezolvareSistem(A,n,B,X));
  readln();
end.
```

Problema 3: Jocuri de minime

Enunţ

Se citeşte o matrice pătratică cu *n* linii şi *n* coloane ($3 \square n \square 100$), 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) Interschimbaţi circular în sens trigonometric (invers acelor de ceas) cele 4 valori minime, de ori câte ori ar apărea valoarea înlocuită în zona de destinaţie. Astfel, valoarea minimă din zona N va ajunge pe poziţiile 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. Afişaţi matricea rezultată.

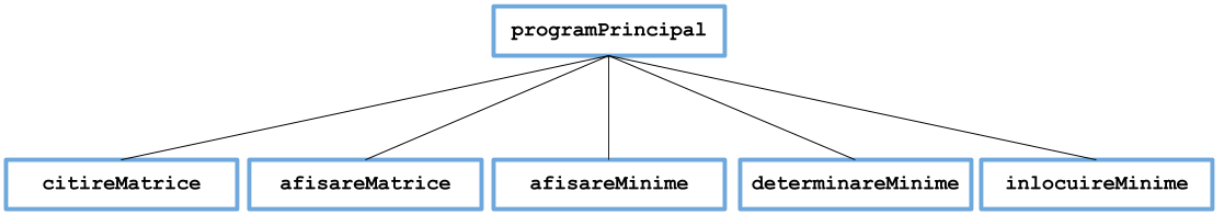
Exemplu

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

Paşii algorimului principal

```
Algoritm rotatiiMinime
     @ citeşte matricea
     @ determina valorile minime din cele patru zone
     @ afişează valorile minime
     @ roteşte valorile minime intre zone
     @ afiseaza matricea noua
Sf.Algoritm
```

Identificarea subalgoritmilor



Programul

```
program problema3;

type
  matrix=array[1..100,1..100] of integer;
  sir=array[1..4] of integer;
var
  matrice:matrix;
  n:integer;
  minime:sir;

procedure citireMatrice(var a:matrix; var dimensiune:integer);
var
  i,j:integer;
begin
  write('Introduceti dimensiunea matricii n='); readln(dimensiune);
  writeln('Introduceti elementele matricii linie cu linie');
  for i:=1 to dimensiune do
  begin
      for j:=1 to dimensiune do
          read(a[i][j]);
      readln();
  end;
end;

procedure afisareMatrice(a:matrix; dimensiune:integer);
var
  i,j:integer;

begin
  writeln();
  for i:=1 to dimensiune do
      begin
          for j:=1 to dimensiune do
              write(a[i][j],' ');
          writeln();
      end;
end;

procedure afisareMinime(s:sir);
begin
  writeln('minim N:', s[1]);
```

```
  writeln('minim V:', s[2]);
  writeln('minim S:', s[3]);
  writeln('minim E:', s[4]);
end;

procedure determinareMinime(a:matrix; dimensiune:integer;var
sirMinime:sir);
// determina cele patru valori minime N,V,S,E

var
  i,j:integer;
begin
  for i:=1 to 4 do
      sirMinime[i]:=1001;
  for i:=1 to dimensiune do
      for j:=1 to dimensiune do
          begin
          if(i<j) and (i+j<dimensiune+1) and (a[i][j]<sirMinime[1])
then
                      sirMinime[1]:=a[i][j];
          if(i<j) and (i+j>dimensiune+1) and (a[i][j]<sirMinime[2])
then
                      sirMinime[2]:=a[i][j];
          if(i>j) and (i+j>dimensiune+1) and (a[i][j]<sirMinime[3])
then
                      sirMinime[3]:=a[i][j];
          if(i>j) and (i+j<dimensiune+1) and (a[i][j]<sirMinime[4])
then
                      sirMinime[4]:=a[i][j];
          end;
end;

procedure inlocuireMinime(var a:matrix;
dimensiune:integer;sirMinime:sir);
// inlocuieste valorile intre ele conform cerintei

var
  i,j:integer;
begin
  for i:=1 to dimensiune do
      for j:=1 to dimensiune do
          begin
          if(i<j) and (i+j<dimensiune+1) and (a[i][j]=sirMinime[1])
then
                      a[i][j]:=sirMinime[2];
          if(i<j) and (i+j>dimensiune+1) and (a[i][j]=sirMinime[2])
then
                      a[i][j]:=sirMinime[3];
          if(i>j) and (i+j>dimensiune+1) and (a[i][j]=sirMinime[3])
then
                      a[i][j]:=sirMinime[4];
          if(i>j) and (i+j<dimensiune+1) and (a[i][j]=sirMinime[4])
then
```

```
                    a[i][j]:=sirMinime[1];
          end;
end;

begin
  citireMatrice(matrice,n);
  determinareMinime(matrice,n,minime);
  afisareMinime(minime);
  inlocuireMinime(matrice,n,minime);
  afisareMatrice(matrice,n);
  readln();
end.
```

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 să 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 x m* ce va conţine doar zero şi unu precum şi matricea *R* ce o va folosi pentru rotire (de dimensiunea *2 x 2* ce conţine patru numere reale - coeficienţii transformării în vederea rotirii).

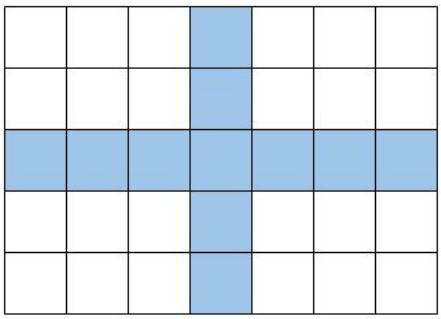
Pentru fiecare punct negru (1 în matrice) se calculează coordonatele iniţiale (x_i, y_i), unde x_i e linia minus $n \; div \; 2 + n \; mod \; 2$ şi y_i este respectiv coloana minus $m \; div \; 2 + n \; mod \; 2$ (pentru a stabili originea sistemului de coordonate în mijlocul matricii).

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

Exemplu

Date de intrare	Rezultate
Imaginea:	

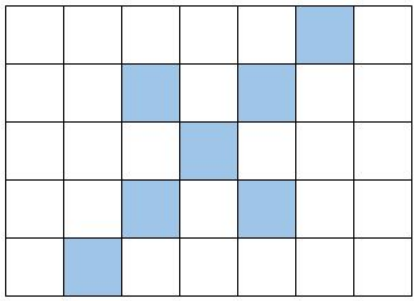
n=5
m=7



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:

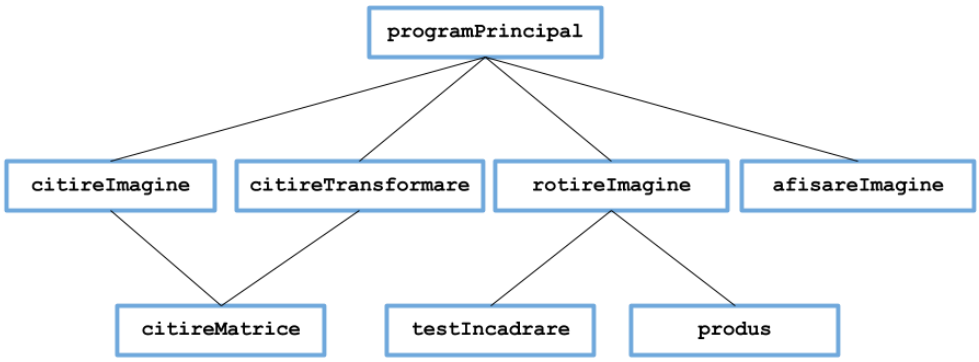


0 0 0 0 0 1 0
0 0 1 0 1 0 0
0 0 0 1 0 0 0
0 0 1 0 1 0 0
0 1 0 0 0 0 0

Paşii algorimului principal

```
Algoritm joculCuImagini
     @ citeşte 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

```
program problema4;

type
  matrix=record
        a:array[1..100,1..100] of real;
        n,m:integer;
        end;

var
  img, rot, imgNew:matrix;

procedure citireMatrice(var mat:matrix);

var
   i,j:integer;
begin
   for i:=1 to mat.n do
     begin
        for j:=1 to mat.m do
                read(mat.a[i][j]);
        readln();
     end;
end;

procedure citireImagine(var imagine:matrix);
begin
   write('Introduceti inaltimea imaginii n='); readln(imagine.n);
   write('Introduceti latimea imaginii m=');readln(imagine.m);
   writeln('Introduceti valoarea punctelor (0 sau 1) linie cu line:');
   citireMatrice(imagine);
end;

procedure citireTransformare(var rot:matrix);
begin
   rot.n:=2;
   rot.m:=2;
    writeln('Introduceti  matricea  transformarii  cu  dimensiunea  2  x  2
```

```
linie cu linie');
   citireMatrice(rot);
end;

procedure produs(m1:matrix; m2:matrix; var c:matrix);
// calculeaza produsul matricilor m1 si m2. Rezultatul se pune in c
var
   i,j,k:integer;
begin
   if m1.m<>m2.n then
        begin
           c.n:=0;
           c.m:=0;
        end
     else
        begin
           c.n:=m1.n;
           c.m:=m2.m;
           for i:=1 to m1.n do
                for j:=1 to m2.m do
                   begin
                      c.a[i][j]:=0;
                      for k:=1 to m1.m do
                        c.a[i][j]:=c.a[i][j]+m1.a[i][k]*m2.a[k][j];
                   end;
        end;
end;

function testIncadrare(x,y,n,m:integer):boolean;
//testeaza daca indicii x si y sunt in intevalele corecte

begin
   testIncadrare:=(x>=1)and(x<=n)and(y>=1)and(y<=m);
end;

procedure rotireImagine(img:matrix; R:matrix; var newImg:matrix);
//roteste imaginea conform coeficientilor din R

var
   coordInitiale, coordTransf:matrix;

   i,j, x,y:integer;
   ajustareX,ajustareY:integer;

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

   ajustareX:=img.n div 2+img.n mod 2;
   ajustareY:=img.m div 2+img.m mod 2;
```

```
   for i:=1 to newImg.n do
        for j:=1 to newImg.m do
            newImg.a[i][j]:=0;
   for i:=1 to img.n do
        for j:=1 to img.m do
            if img.a[i][j]=1 then
                begin
                  coordInitiale.a[1][1]:=i-ajustareX;
                  coordInitiale.a[2][1]:=j-ajustareY;

                  produs(R, coordInitiale, coordTransf);

                  x:=trunc(coordTransf.a[1][1])+ajustareX;
                  y:=trunc(coordTransf.a[2][1])+ajustareY;
                  write(x,' ',y,' ');

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

procedure afisareImagine(imag:matrix);

var
   i,j:integer;

begin
   for i:=1 to imag.n do
       begin
          writeln();
          for j:=1 to imag.m do
             write(imag.a[i][j]:1:0 , ' ');
       end;
end;

begin
   citireImagine(img);
   citireTransformare(rot);
   rotireImagine(img,rot,imgNew);
   afisareImagine(imgNew);
   readln();
end.
```