ALGORITMI ELEMENTARI FOLOSIND MATRICE (continuare)

Zona de ***deasupra diagonalei principale*** a unei matrice pătrate de dimensiune *n* cu declarată în C/C++ cu indicii liniilor și coloanelor de la 0 la *n*-1 conține elementele adică elementele **(*a*[*i*][*j*]) cu *i* = 0, 1,..., *n*-2 și *j* = *i*+1, ..., *n*-1.** Ar fi mai puțin de preferat să se parcurgă întreaga matrice (*i*, *j* = 0, ..., *n*-1 și apoi să se testeze dacă *i* < *j* pentru zona de deasupra diagonalei principale).

$$\begin{pmatrix} a[0][0] & \mathbf{a[0][1]} & \mathbf{a[0][2]} & \dots & \dots & \mathbf{a[0][n-1]} \\ & a[1][1] & \mathbf{a[1][2]} & \dots & \dots & \mathbf{a[1][n-1]} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & & a[i][i] & \mathbf{a[i][i+1]} & \mathbf{a[i][n-1]} \\ & & & & \dots & \\ \dots & \dots & \dots & \dots & a[n-2][n-2] & \mathbf{a[n-2][n-1]} \\ & & & & & a[n-1][n-1] \end{pmatrix},$$

Zona de ***sub diagonala principală*** a unei matrice pătrate de dimensiune *n* cu declarată în C/C++ cu indicii liniilor și coloanelor de la 0 la *n*-1 conține elementele adică elementele **(*a*[*i*][*j*]) cu *i* = 1, ..., *n*-1 și *j* = 0, ..., *i*-1.** Ar fi mai puțin de preferat să se parcurgă întreaga matrice (*i*, *j* = 0, ..., *n*-1 și apoi să se testeze dacă *i*>*j* pentru zona de sub diagonala principală.

$$\begin{pmatrix} a[0][0] & & & & & & \\ \mathbf{a[1][0]} & a[1][1] & & & & & \\ \mathbf{a[2][0]} & \mathbf{a[2][1]} & a[2][2] & \dots & \dots & . & \dots \\ \dots & \dots & \dots & \dots & & & \\ \mathbf{a[i][0]} & \mathbf{a[i][1]} & \dots & \mathbf{a[i][i-1]} & a[i][i] & \dots & \\ \dots & \dots & \dots & \dots & \dots & a[n-2][n-2] & \\ \mathbf{a[n-1][0]} & \mathbf{a[n-1][1]} & & & & \mathbf{a[n-1][n-2]} & a[n-1][n-1] \end{pmatrix}$$

Lucru individual:

Stabiliţi dacă o matrice este simetrică sau nu față de diagonala principală (adică, *A* este simetrică dacă $a_{ij} = a_{ji}$, pentru orice $i \neq j$). Sugestie: se pot parcurge elementele de deasupra diagonalei principale (sau cele de sub diagonala principală, dar nu ambele căci se fac comparații duble) şi dacă elementul curent *a*[*i*][*j*] este diferit de simetricul său *a*[*j*][*i*] atunci este clar că matricea nu este simetrică; ar fi preferată folosirea unei funcții de genul:

```
bool MatriceSimetrica (float a[10][10], int n)
{
  for (int i=0; i<=n-2; i++) //deasupra diagonalei princ.
    for (int j=i+1; j<=n-1; j++)
        if (a[i][j] != a[j][i]) return false;
        return true;
}
```

R1 (4*).

Enunţul problemei: Să se descrie un algoritm pentru a determina media aritmetică a elementelor de deasupra diagonalei principale.

Metoda de rezolvare: De exemplu, pentru $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ media aritmetică a elementelor de deasupra diagonalei principale este: $\frac{2+3+6}{3} = \frac{11}{3} = 3,(6)$.

În C/C++:

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots & \dots & \dots & \dots & \dots & a_{0,n-2} & a_{0,n-1} \\ a_{10} & a_{11} & a_{12} & \dots & \dots & \dots & \dots & \dots & a_{1n-2} & a_{1,n-1} \\ a_{20} & a_{21} & a_{13} & \dots & \dots & \dots & \dots & \dots & a_{2,n-2} & a_{2,n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i0} & a_{i1} & a_{i2} & \dots & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \dots & a_{i,n-2} & a_{i,n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n-2,0} & a_{n-2,1} & a_{n-2,2} & \dots & \dots & \dots & \dots & \dots & a_{n-2,n-2} & a_{n-2,n-1} \\ a_{n-1,0} & a_{n-1,1} & a_{n-1,2} & \dots & \dots & \dots & \dots & \dots & a_{n-1,n-2} & a_{n-1,n-1} \end{pmatrix}$$

elementele de deasupra diagonalei principale sunt $(a_{ij})_{i = 0, 1,\dots, n-2;\ j = i+1, n-1}$.

Numărul elementelor de pe diagonala principală se poate determina cu un contor pe parcurs ce se parcurge zona de deasupra diagonalei principale sau se poate determina matematic: $(n–1) + (n–2) + \dots + 1 = (n–1)(n–1+1) / 2 = n(n–1)/2$. În acest ultim caz, rămâne doar să parcurgem zona de deasupra diagonalei principale şi să însumăm elementele, apoi media aritmetică reprezintă raportul dintre suma elementelor şi numărul lor.

O descriere a algoritmului în pseudocod:

```
citeşte n,a00,…,an-1,n-1
S ← 0 *initial, suma elementelor de deasupra diag.princ. este 0
pentru i = 0,n-2 repeta     *se parcurge zona de deasupra diag.
  pentru j = i+1,n-1 repeta* principale
     S ← S + aij            *insumam elementul curent
scrie  S / (n(n-1)/2)      *se afiseaza valoarea mediei aritmetice
```

O descriere a algoritmului în pseudocod C++ (CodeBlocks):

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
 int n,i,j;
 float a[10][10],S;

 cout<<"Dati dimensiunea matricei: "; cin>>n;
 cout<<"Dati elementele matricei: "<<endl;
 for (i=0;i<n;i++)
   for (j=0;j<n;j++) cin>>a[i][j];

 S = 0;    //numarul elem. de deasupra diagonalei principale
 for (i=0;i<=n-2;i++)       //se parcurge zona de deasupra
  for (j=i+1;j<=n-1;j++)    //diagonalei principale
    S += a[i][j];
```

```
 cout<<"Media aritmetica a elementelor de deasupra diagonalei
principale este: "<<fixed<<setprecision(2) << S / (n*(n-1)/2)
<<endl;
 return 0;
}
```

Rulare:

```
Dati dimensiunea matricei: 3 <Enter>
Dati elementele matricei:
1 2 3 <Enter>
4 5 6 <Enter>
7 8 9 <Enter>
Media aritmetica a elementelor de deasupra diagonalei principale
este: 3.67
```

Lucru individual:

Determinați valoarea medie (media aritmetică) a elementelor din zona de deasupra diagonalei principale, plus diagonala principală. Sugestie: se parcurg elementele de **deasupra diagonalei principale și de pe diagonala principală (*i* = 0,..., *n*-1, iar *j* = *i*,..., *n*-1)** și se însumează (eventual se pot și contoriza); la final se împarte suma lor la numărul lor (a cărei valoare se determină cu suma lui Gauss: *n* (elem pe linia 0) + *n*-1 (elemente pe linia 1) +… + 2 (elemente pe linia *n*-2) + 1 (element pe linia *n*-1) = *n*(*n*+1)/2.

R2 (4*).

Enunţul problemei: Verificaţi dacă o matrice pătrată este superior triunghiulară.

Metoda de rezolvare: O matrice este superior triunghiulară dacă toate elementele de sub diagonala principală sunt nule (adică $a_{ij} = 0$, pentru $i > j$ şi elementele esenţiale sunt pe şi deasupra diagonalei principale). Un exemplu de matrice superior triunghiulară este $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{pmatrix}$.

Descrierea algoritmului în pseudocod C++ (CodeBlocks):

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots & a_{0i} & \dots & a_{0n} \\ 0 & a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1n} \\ 0 & 0 & a_{22} & \dots & a_{2i} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & a_{ii} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 0 & \dots & a_{n-1,n-1} \end{pmatrix}$$, adică

$a_{ij} = 0$, pentru orice ***i* = 1...*n*-1** şi ***j* = 1,..., *i*-1** (**sub diagonala principală** numai 0).

O descriere a algoritmului în pseudocod C++ (CodeBlocks):

```
#include <iostream>
using namespace std;
int main()
{
 int n,i,j,OK; //"stil Pascalist"
 float a[10][10];
```

```
 cout<<"Dati dimeniunea matricei: "; cin>>n;
 /*for (i=0;i<n;i++)
    for (j=0;j<n;j++){cout<<"a["<<i+1<<"]["<<j+1<<"]=";
                      cin>>a[i][j]; } */   // sau
 cout<<"Dati elementele matricei:"<<endl;
 for (i=0;i<n;i++)
  for (j=0;j<n;j++) cin>>a[i][j];
 OK = 1;   //pp. ca matricea este superior triunghiulara
 for (i=1;i<n;i++) //parcurg elem de sub diag principala
  for (j=0;j<i;j++)
   if (a[i][j]) //sau if (a[i][j]!=0)
   {OK = 0;
    //i = n; pregatim iesirea si din for-ul dupa i
    break; // iese din for-ul dupa j
   }
 if (OK)   //sau if (OK==1)
    cout<<"Matricea data este superior triunghiulara"<<endl;
 else
    cout<<"Matricea data nu este superior triunghiulara"<<endl;
 return 0;
}
```

sau folosind funcţii (de preferat):

```
#include <iostream>
using namespace std;
void Citire(float a[10][10], int &n)
{
 cout<<"Dati dimeniunea matricei: "; cin>>n;
 cout<<"Dati elementele matricei:"<<endl;
 for (int i=0;i<n;i++)
  for (int j=0;j<n;j++) cin>>a[i][j];
}
int EsteSuperiorTriunghiulara(float a[10][10], int n)
{
 for (int i=1;i<n;i++) //parcurg elem de sub diag principala
  for (int j=0;j<i;j++)
   if (a[i][j]!=0) //sau if (a[i][j])
    return 0;
 return 1;
}
int main()
{int n;
 float a[10][10];
 Citire(a,n);
 if (EsteSuperiorTriunghiulara(a,n))
    cout<<"Matricea data este superior triunghiulara"<<endl;
 else
    cout<<"Matricea data nu este superior triunghiulara"<<endl;
 return 0;
}
```

Rulare:

```
Dati dimeniunea matricei: 4 <Enter>
Dati elementele matricei:
1 1 1 0
0 2 1 1
0 0 3 1
0 0 0 5
```

```
      Matricea data este superior triunghiulara
```
sau
```
      Dati dimeniunea matricei: 4 <Enter>
      Dati elementele matricei:
      1 1 1 1
      0 2 1 0
      2 0 3 1
      0 1 0 5
      Matricea data nu este superior triunghiulara
```

Elementele de pe ***diagonala secundară*** a unei matrice pătrate de dimensiune *n* sunt **(*a*[*i*][*n*-1-*i*]), *i* = 0, ..., *n*-1.**

Zona de ***deasupra diagonalei secundare*** a unei matrice pătrate de dimensiune *n* cu declarată în C/C++ cu indicii liniilor și coloanelor de la 0 la *n*-1 conține elementele adică elementele **(*a*[*i*][*j*]) cu *i* = 0, 1,..., *n*-1 și *j* = 0, ..., *n*-2-*i*.** Ar fi mai puțin de preferat să se parcurgă întreaga matrice (*i*, *j* = 0, ..., *n*-1 și apoi să se testeze dacă *i*+*j* < *n*-1 pentru zona de deasupra diagonalei secundare).

$$\begin{pmatrix} a[0][0] & a[0][1] & a[0][2] & \dots & \dots & a[0][n-1] \\ a[1][0] & a[1][1] & a[1][2] & \dots & \dots & a[1][n-1] \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & & a[i][i] & a[i][i+1] & a[i][n-1] \\ & & & & \dots & \\ \dots & \dots & \dots & \dots & \dots & a[n-2][n-1] \\ a[n-1][0] & a[n-1][1] & & & & a[n-1][n-1] \end{pmatrix}$$

Zona de ***sub diagonala secundară*** a unei matrice pătrate de dimensiune *n* cu declarată în C/C++ cu indicii liniilor și coloanelor de la 0 la *n*-1 conține elementele adică elementele **(*a*[*i*][*j*]) cu *i* = 1, ..., *n*-1 și *j* = *n*-*i*, ..., *n*-1.** Ar fi mai puțin de preferat să se parcurgă întreaga matrice (*i*, *j* = 0, ..., *n*-1 și apoi să se testeze dacă *i*+*j*>*n*-1 pentru zona de sub diagonala secundară).

$$\begin{pmatrix} a[0][0] & a[0][1] & & & & & a[0][n-1] \\ a[1][0] & a[1][1] & & & & & a[1][n-1] \\ a[2][0] & a[2][1] & \dots & \dots & . & & \dots \\ \dots & \dots & \dots & \dots & & & \\ a[i][0] & a[i][1] & \dots & a[i][i-1] & a[i][i] & \dots & \\ \dots & \dots & \dots & \dots & \dots & \dots & a[n-2][n-2] \\ a[n-1][0] & a[n-1][1] & & & & a[n-1][n-2] & a[n-1][n-1] \end{pmatrix}$$