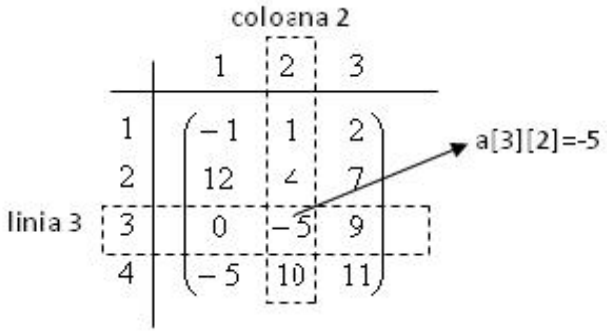
MATRICI(TABLOURI BIDIMENSIONALE)

O matrice reprezintă o succesiune de locaţii de memorie recunoscute prin acelaşi identificator şi prin poziţia fiecăreia în cadrul şirului. Poziţia este dată printr-o suită de două numere pozitive (indecşi), care reprezintă cele două dimensiuni (linie şi coloană).Valorile atribuite elementelor tabloului trebuie să fie de acelaşi tip.

O matrice este un tabel cu elemente de acelaşi tip, dispuse pe linii şi coloane . Poziţia unui element pe linie se numeşte şi indice de linie, iar poziţia elementului pe coloană se mai numeşte şi indice de coloană. Prin a[i][j] înţelegem elementul care se află pe linia i şi coloana j.



Declararea matricei

Sintaxa generală:
tip_nume [marime1][marime2], (în C++ matricea fiind de fapt un vector de vectori) , unde:
- tip – tipul de bază al componentelor matricei;
- nume – numele matricei;
- marime 1 – numărul de linii al matricei;
- marime 2 – numărul de coloane al matricei;

Exemplu: int a[7][8];
am declarat o matrice care are 7 linii şi 8 coloane , numerotate de la 0 la 6 , respectiv de la 0 la 7 (există posibilitatea , chiar indicată , de a se numerota de la 1 la 7 ,respectiv de la 1 la 8).
Fiecare element al matricei este de tipul int. Ele se adresează astfel : a[0][0],a[0][1] . . . a[6][7].

Limbajul C++ nu ne permite să declarăm o variabilă tablou cu număr variabil de componente , posibilitate oferită de alte limbaje. De multe ori nu se cunoaşte numărul de componente necesar rulării programului. Este necesar să rezervăm un număr maxim de componente , atât cât este necesar pentru rulare când n este maxim. La fiecare rulare a programului i se cere numărul de componente. De cele mai multe ori , o parte din ele rămân neutilizate. În memorie , tablourile sunt memorate pe linii. Asta inseamnă că la început este memorată prima linie , apoi a doua şi aşa mai departe .

Citirea matricei

Citirea elementelor unui tablou nu este posibilă decât prin citirea fiecărui element . De aceea, operaţia de citire a matricelor impune folosirea a două secvenţe ciclice suprapuse. Acestea corespund indicelor liniei (i) , respectiv coloanei (j).
Citirea matricei se face având următoarea sintaxă generală:

```
cout<< ”n= ”;
cin>>n;
for ( i=1; i<=m; i++)
for ( j=1; j<=n; j++)
{ cout<< ”a [”<<i<< ” , ”<<j<< ”] = ”;
cin>>a[i][j];

}
```

Afişarea matricei

Afişarea matricei se face având urmatoarea sintaxă :

```
for ( i=1; i<=m; i++)
  { for ( j=1; j<=n; j++)
          cout<<a[i][j]<<’ ’;
    cout<<endl;
  }
```

Parcurgerea unei matrici

O matrice poate fi parcursă pe linii de la prima linie către ultima sau invers iar in cadrul fiecărei linii de la stânga la dreapta sau de la dreapta la stânga, respectiv pe coloane de la prima coloană către ultima sau invers iar în cadrul fiecărei coloane de sus în jos sau de jos în sus. Există și alte posibilități de parcurgere: pe chenere concentrice, sub formă de L, șerpuit, fiecare modalitate respectând anumite reguli prin care se acceseayă elementele în ordinea dorită.

1. Parcurgerea matricii pe linii de la prima către ultima și pe fiecare linie de la stânga la dreapta

```
for ( i=1; i<=m; i++)
   for ( j=1; j<=n; j++)
          prelucrează(a[i][j]);
```

2. Parcurgerea matricii pe linii de la prima către ultima și pe fiecare linie de la dreapta la stânga

```
for ( i=1; i<=m; i++)
   for ( j=n; j>=1; j--)
          prelucrează(a[i][j]);
```

3. Parcurgerea matricii pe linii de la ultima către prima și pe fiecare linie de la stânga la dreapta

```
for ( i=m; i>=1; i--)
   for ( j=1; j<=n; j++)
          prelucrează(a[i][j]);
```

4. Parcurgerea matricii pe linii de la ultima către prima și pe fiecare linie de la dreapta la stânga

```
for ( i=m; i>=1; i--)
   for ( j=n; j>=1; j--)
          prelucrează(a[i][j]);
```

5. Parcurgerea matricii pe coloane de la prima către ultima și pe fiecare coloană de sus în jos

```
for ( j=1; j<=n; j++)
```

```
for ( i=1; i<=m; i++)
        prelucrează(a[i][j]);
```

6. Parcurgerea matricii pe coloane de la prima către ultima și pe fiecare coloană de jos în sus

```
for ( j=1; j<=n; j++)
    for ( i=m; i>=1; i--)
        prelucrează(a[i][j]);
```

7. Parcurgerea matricii pe coloane de la ultima către prima și pe fiecare coloană de sus în jos

```
for ( j=n; j>=1; j--)
    for ( i=1; i<=m; i++)
        prelucrează(a[i][j]);
```

8. Parcurgerea matricii pe coloane de la ultima către prima și pe fiecare coloană de jos în sus

```
for ( j=n; j>=1; j--)
    for ( i=m; i>=1; i--)
        prelucrează(a[i][j]);
```

Schema generală de prelucrare a elementelor unei matrici atunci când o parcurgem de la prima linie către ultima și pe fiecare linie de la stânga la dreapta poate avea forma:

```
inițializare calcule pentru prelucrări referitoare la întreaga matrice
  for ( i=1; i<=m; i++)
   { inițializare calcule pentru prelucrări referitoare la elementele de pe linia i
    for ( j=1; j<=n; j++)
        {
        prelucrează(a[i][j]);
         }
    interpretare / afișare rezultate calcule obținute pentru linia i
    }
Interpretare/ afișare rezultate calcule obținute pentru întreaga matrice
```

Matrici pătrate

Sunt acele matrici în care numărul de linii este egal cu numărul de coloane.
Având formă de pătrat putem vorbi despre diagonale:

- o diagonală principală, cea care este formată din elementele situate începând cu cel din colțul din stânga sus și până la cel situat în colțul din dreapta jos
- o diagonală secundară, cea care este formată din elementele situate începând cu cel din colțul din dreapta sus și până la cel situat în colțul din stânga jos

Diagonala principală:

a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]
a[3][1]	a[3][2]	a[3][3]	a[3][4]	a[3][5]
a[4][1]	a[4][2]	a[4][3]	a[4][4]	a[4][5]
a[5][1]	a[5][2]	a[5][3]	a[5][4]	a[5][5]

Elementele a[i]j] situate pe diagonala principală îndeplinesc condiția **i==j.**
Elementele a[i]j] situate **deasupra** diagonalei principale îndeplinesc condiția **i<j**.
Elementele a[i]j] situate **sub** diagonala principală îndeplinesc condiția **i>j**.

```
for ( i=1; i<=n; i++)
   for ( j=1; j<=n; j++)
          if(i==j) cout<<a[i][j]<<"pe diagonala principala";
          else if(i<j) cout<<a[i][j]<<"deasupra diagonalei principale";
               else cout<<a[i][j]<<"sub diagonala principala";
```

Simetria față de diagonala principală: a[i][j] simetric cu a[j][i].
Paralele la diagonala principală:
Pentru paralela k de deasupra sau sub diagonala principală vom avea
abs(i-j)==k

0	1	2	3	4
1	0	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	1	4

Diagonala secundară:

a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]
a[3][1]	a[3][2]	a[3][3]	a[3][4]	a[3][5]
a[4][1]	a[4][2]	a[4][3]	a[4][4]	a[4][5]
a[5][1]	a[5][2]	a[5][3]	a[5][4]	a[5][5]

Elementele a[i]j] situate pe diagonala secundară îndeplinesc condiția **i+j==n+1.**
Elementele a[i]j] situate deasupra diagonalei secundare îndeplinesc condiția **i+j<n+1**.
Elementele a[i]j] situate sub diagonala secundară îndeplinesc condiția **i+j>n+1.**

Observație: dacă elementele matricii se consideră situate începând cu linia 0 și coloana 0 atunci pentru diagonala secundară ne vor referi la n-1 în loc de n+1.

```
for ( i=1; i<=n; i++)
   for ( j=1; j<=n; j++)
          if(i+j==n+1) cout<<a[i][j]<<"pe diagonala secundara";
          else if(i+j<n+1) cout<<a[i][j]<<"deasupra diagonalei secundare";
               else cout<<a[i][j]<<"sub diagonala secundara";
```

Vom compara i+j cu n-1 dacă numerotăm liniile și coloanele de la 0.
Simetria față de diagonala secundară: **a[i][j]** simetric cu **a[n-j+1][n-i+1]** sau dacă numerotăm liniile și coloanele de la 0: a[i][j] simetric cu a[n-j-1][n-i-1]
Paralele la diagonala secundară:

Pentru paralela k de deasupra sau sub diagonala secundară vom avea
abs(n+1-(i+j))==k
sau dacă numerotăm liniile și coloanele de la 0:
abs(n-1-(i+j))

4	3	2	1	0
3	2	1	0	1
2	1	0	1	2
1	0	1	2	3
0	1	2	3	4

Observație: Regulile vor fi adaptate în funcție de cerințele problemei.

Vecinii unui element:

a[i-1][j-1]	a[i-1][j]	a[i-1][j+1]
a[i][j-1]	a[i][j]	a[i][j+1]
a[i][j-1]	a[i+1][j]	a[i+1][j+1]

Un element a[i][j] poate avea cel mult 8 vecini direcți situați pe linia anterioară sau pe linia următoare și respectiv pe o coloană mai la stânga sau mai la dreapta.

Elementele situate pe chenarul matricii au mai puțini vecini direcți în funcție de poziția lor(cei de pe prima linie nu au vecinii de pe linia anterioară, cei de pe prima coloană nu au vecinii de pe coloana anterioară etc.)

Simetria în cadrul liniei i

- **pentru elemente egal depărtate de capete**

1	**2**	**3**	**4**	**4**	**3**	**2**	**1**

for(j=1;j<=n/2;j++)

a[i][j] simetric cu a[i][n-j+1] pentru numerotarea de la 1

- **pentru fiecare în ordine din jumătatea lui (nr elem par)**

1	**2**	**3**	**4**	**1**	**2**	**3**	**4**

for(j=1;j<=n/2;j++)

a[i][j] simetric cu a[i][n/2+j]

Construirea unei matrici:

Presupune citirea(dacă se cunoaște) a numărului de linii și coloane ale matricii și apoi parcurgerea acesteia după anumite reguli pentru completarea elementelor cu valori. După construire matricea se afișează pentru a putea astfel verifica dacă a fost construită corect.

Observație: În funcție de regula de completare a elementelor se vor scri instrucțiunile de parcurgere a matricii. De exemplu dacă un element se construiește în funcție de vecinii de pe linia anterioară liniile se vor parcurge de la prima către ultima, dar dacă depind de elemnentele de pe linia următoare se vor construi de la ultima linie către prima. Analog, dacă un element depinde de elementele de pe coloana din stânga(anterioară) coloanele se vor parcurge de la prima către ultima, dar dacă depinde de elementele de pe coloana din drepata (următoarea) coloanele se vor parcurge de la ultima către prima etc.

Problemă: Scrieţi un program C/C++ care citeşte de la tastatură două numere naturale nenule n şi m (2≤m≤10, 2≤n≤10) şi care construieşte în memorie şi apoi afişează o matrice a cu n linii (numerotate de la 1 la n) şi m coloane (numerotate de la 1 la m) cu proprietatea că fiecare element a[i][j] memorează cea mai mică dintre valorile indicilor i şi j (1≤i≤n, 1≤j≤m). Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spaţiu.

Exemplu: pentru n=4 şi m=5 se va afişa matricea alăturată.
1 1 1 1 1
1 2 2 2 2
1 2 3 3 3
1 2 3 4 4

```
#include<iostream>
using namespace std;
int main()
{int a[11][11],n,m,i,j;
cout<<"n=";cin>>n;
cout<<"m=";cin>>m;
for ( i=1; i<=n; i++)
    for ( j=1; j<=m; j++)
       if(i<j) a[i][j]=i;
       else a[i][j]=j;
for ( i=1; i<=n; i++)
  { for ( j=1; j<=nm j++)
            cout<<a[i][j];
     cout<<endl;
  }
return 0;
}
```

Problemă: Scrieţi un program C/C++ care citeşte de la tastatură un număr natural n (2≤n≤24) şi construieşte în memorie o matrice cu n linii şi n coloane ale cărei elemente vor primi valori după cum urmează: - elementele aflate pe diagonala principală a matricei vor primi valoarea 0 - elementele de pe prima coloană, cu excepţia celui aflat pe diagonala principală vor primi valoarea n - elementele de pe a doua coloană, cu excepţia celui aflat pe diagonala principală vor primi valoarea n-1 ... - elementele de pe ultima coloană, cu excepţia celui aflat pe diagonala principală vor primi valoarea 1 Programul va afişa matricea astfel construită pe ecran, câte o

linie a matricei pe câte o linie a ecranului, cu câte un spaţiu între elementele fiecărei linii (ca în exemplu).

Exemplu: pentru n=4 se va afişa matricea alăturată.

0 3 2 1
4 0 2 1
4 3 0 1
4 3 2 0

```
#include<iostream>
using namespace std;
int main()
{int a[25][25],n,i,j;
cout<<"n=";cin>>n;
for ( i=1; i<=n; i++)
   for ( j=1; j<=n; j++)
      if(i==j) a[i][j]=1;
      else a[i][j]=n-j+1;
for ( i=1; i<=m; i++)
  { for ( j=1; j<=n; j++)
            cout<<a[i][j];
    cout<<endl;
  }
return 0;
}
```

Problemă : Scrieţi un program C/C++ care citeşte de la tastatură două numere naturale n şi p (2≤n≤20, 1≤p≤20) şi construieşte în memorie un tablou bidimensional cu n linii şi p coloane. Tabloul va fi construit astfel încât, parcurgând tabloul linie cu linie de sus în jos şi fiecare linie de la stânga la dreapta, să se obţină şirul primelor n*p pătrate perfecte impare, ordonat strict crescător, ca în exemplu. Tabloul astfel construit va fi afişat pe ecran, fiecare linie a tabloului pe câte o linie a ecranului, cu câte un spaţiu între elementele fiecărei linii. **Exemplu**: pentru n=2, p=3 se va afişa tabloul alăturat:

```
1    9    25
49   81   121
```

```
#include<iostream>
using namespace std;
int main()
{int a[21][21],n,p,i,j,k;
cout<<"n=";cin>>n;
cout<<"p=";cin>>p;
k=1;
for ( i=1; i<=n; i++)
   for ( j=1; j<=p; j++)
      { a[i][j]=k*k;
        k=k+2;
      }
for ( i=1; i<=n; i++)
  { for ( j=1; j<=p; j++)
            cout<<a[i][j];
```

```
    cout<<endl;
  }
return 0;
}
```

Problemă: Se consideră tabloul bidimensional cu n linii şi n coloane ce conţine numere naturale cu cel mult patru cifre fiecare. Scrieţi programul C/C++ care citeşte de la tastatură numărul natural n (2≤n≤23) şi cele n*n elemente ale tabloului şi apoi afişează pe ecran elementele primului pătrat concentric, separate prin câte un spaţiu. Pătratul este parcurs în sensul acelor de ceasornic începând din colţul său stânga-sus, ca în exemplu. Primul pătrat concentric este format din prima şi ultima linie, prima şi ultima coloană a tabloului.
Exemplu: pentru n=5 şi tabloul alăturat, se va afişa: 1 2 3 4 5 1 6 2 7 6 5 4 3 7 2 6

1	2	3	4	5
6	7	8	9	1
2	3	4	5	6
7	8	9	1	2
3	4	5	6	7

```
#include<iostream>
using namespace std;
int main()
{int a[24][24],n,i,j;
cout<<"n=";cin>>n;
cout<<"m=";cin>>m;
for ( i=1; i<=n; i++)
for ( j=1; j<=m;
j++)
       cin>>a[i][j];
for(j=1;j<=n;j++) cout<<a[1][j]<<' ';    //prima linie de la stânga la dreapta
for(i=2;i<=n;i++) cout<<a[i][n]<<' ';    // ultima coloană de sus în jos fără primul element
for(j=n-1;j>=1;j--) cout<<a[n][j]<<' ';  //ultima linie de la dreapta la stânga fără ultimul element
for(i=n-1;i>=2;i--) cout<<a[i][1]<<' ';  //prima coloană de jos în sus fără ultimul element
return 0;
}
```

Problemă(Parcurgerea în spirală): Scrieţi un program C/C++ care citeşte de la tastatură un număr natural nenul n (n≤10) și construiește o matrice cu n linii și n coloane ce conține toate valorile de la 1 la n parcurgând matricea în spirală ca în exemplu.

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

```
#include<iostream>
using namespace std;
int main()
{int a[11][11],n,i,j,k=1,p;
 cout<<"n=";cin>>n;
 for ( p=1; p<=n/2; p++)
   {
    for(j=p;<=n-p+1;j++) {a[p][j]=k ;k++ ;}          //linia p de la stânga la dreapta
    for(i=p+1;i<=n-p+1;i++) {a[i][n-p+1]=k;k++;}     //coloana n-p+1 de sus în jos
    for(j=n-p;j>=p;j--) {a[n-p+1][j]=k ;k++ ;}       //linia n-p+1 de la dreapta la stânga
    for(i=n-p;i>=p+1;i--) {a[i][p]=k;k++ ;}          //coloana p de jos în sus
   }
  if(n%2 !=0) a[n/2+1][n/2+1]=k ;  //cazul n impar rămâne un element în mijloc
for ( i=1; i<=n; i++)
  { for ( j=1; j<=p; j++)
           cout<<a[i][j];
```

```
      cout<<endl;
   }
return 0;
}
```

Problemă: Scrieţi un program C/C++ care citeşte de la tastatură două valori naturale nenule m şi n (m≤10, n≤10) şi apoi m*n numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu m linii şi n coloane. Programul determină apoi valorile minime de pe fiecare linie a matricei şi afişează pe ecran cea mai mare valoare dintre aceste minime.
Exemplu: pentru m=3, n=5 şi matricea
5 3 13 7 5
9 10 6 12 9
4 7 6 5 2
se afişează pe ecran valoarea 6 (cea mai mică valoare de pe prima linie a matricei este 3, cea mai mică valoare de pe linia a doua este 6, cea mai mică valoare de pe linia a treia este 2. Cea mai mare dintre aceste trei valori este 6).

```
#include<iostream>
using namespace std;
int main()
{int a[11][11],n,i,j,minl,maxa;
cout<<"m=";cin>>m;
cout<<"n=";cin>>n;
for ( i=1; i<=m; i++)
    for ( j=1; j<=n; j++)
        cin>>a[i][j];
maxa=a[1][1];            //inițializăm maximul cu primul element din matrice
for ( i=1; i<=m; i++)
{minl=a[i][1];           //inițializăm minimul de pe linia i cu primul element de pe linie
    for ( j=1; j<=n; j++ )
        if(a[i][j]<minl)
            minl=a[i][j];
 if(minl>maxa) maxa=minl;  // prelucrăm minimul liniei i comparându-l cu maximul din minime
}
cout<<maxa;
return 0;
}
```

Problemă: Scrieţi un program C/C++ care citeşte de la tastatură două valori naturale nenule m şi n (m≤10, n≤10) şi apoi m*n numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu m linii şi n coloane. Programul rotește circular la stânga elementele matricii formând o altă matrice pe care apoi o afișează.

```
#include<iostream>
using namespace std;
int main()
{int a[11][11],n,i,j,minl,maxa;
cin>>m; cin>>n;
for ( i=1; i<=m; i++)
    for ( j=1; j<=n; j++)
        cin>>a[i][j];
for ( i=1; i<=m; i++)
    for ( j=1; j<=n; j++)
```

```
        b[j][n-i+1]=a[i][j]
for ( i=1; i<=n; i++)
   {for ( j=1; j<=m; j++)
      cout<<b[i][j]<<' ';
cout<<endl;
}
return 0;
}
```

Ștergerea liniei k

```
for ( i=k+1; i<=m; i++)
   for ( j=1; j<=n; j++)
      a[i-1][j]=a[i][j];
m--;
```

Inserarea unei linii noi după linia k

```
for ( i=m; i>=k+1; i--)
   for ( j=1; j<=n; j++)
      a[i+1][j]=a[i][j];
for(j=1;j<=n;j++)
   a[k+1][j]=valoare de adaugat
m++;
```

Observație: dacă ștergem mai multe linii, la fiecare linie k ștearsă adăugăm k—pentru a sta pe loc, iar la inserare trecem peste k++.

Ordonarea elementelor de pe diagonala principală fără păstrarea elementelor în cadrul fiecărei linii / coloane

4	**3**	**1**	**2**
5	**1**	**6**	**7**
8	**3**	**2**	**1**
9	**4**	**5**	**3**

1	**3**	**1**	**2**
5	**2**	**6**	**7**
8	**3**	**3**	**1**
9	**4**	**5**	**4**

```
for ( i=1; i<n; i++)
   for ( k=i+1; k<=n; k++)
      if(a[i][i]>a[k][k])
            {int aux=a[i][i];a[i][i]=a[k][k];a[k][k]=aux;}
```

Ordonarea elementelor de pe diagonala principală păstrând aceleași elemente în cadrul fiecărei linii / coloane prin interschimbarea liniei i cu linia k și a coloanei i cu coloana k

4	**3**	**1**	**2**

5	**1**	**6**	**7**
8	**3**	**2**	**1**
9	**4**	**5**	**3**

1	**6**	**7**	**5**
3	**2**	**1**	**8**
4	**5**	**3**	**9**
3	**1**	**2**	**4**

```
for ( i=1; i<n; i++)
   for ( k=i+1; k<=n; k++)
      if(a[i][i]>a[k][k])
         {for(j=1;j<=n;j++)
                  {int aux=a[i][j];a[i][j]=a[k][j];a[k][j]=aux;}
          for(i1=1;i1<=n;i1++)
             {int aux=a[i1][i];a[i1][i]=a[i1][k];a[i1][k]=aux;}
         }
```

Rotirea matricii cu 90^0 la dreapta (inițial a are m linii și n coloane matricea b care se formează are n linii și m coloane)

```
for ( i=1; i<=n; i++)
   for ( j=1; j<=m; j++)
      b[i][j]=a[m-j+1][i]
```