INFORMATICĂ · CLASA A X-A · TABLOURI BIDIMENSIONALE · LECȚIA 9

Matrice pătratice: diagonale, zone, simetrii și generări de matrice

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile x-08

CE VEI ȘTI DUPĂ LECȚIE

- Recunoști elementele de pe diagonala principală și cea secundară (i = j, i + j = n + 1).
- Delimitezi cele 4 zone determinate de diagonale (N, S, E, V) cu inegalități.
- Verifici simetria față de diagonală și construiești transpusa.
- Construiești o matrice după o regulă dată (exercițiul de Bac II.3).
- Parcurgi matricea pe chenare, în spirală sau „șerpuit".

1. Matricea pătratică și cele două diagonale

O matrice cu **n linii și n coloane** se numește **pătratică**. Ea are două diagonale importante.

	1	2	3	4	5
1	**1,1**	1,2	1,3	1,4	**1,5**
2	2,1	**2,2**	2,3	**2,4**	2,5
3	3,1	3,2	**3,3**	3,4	3,5
4	4,1	**4,2**	4,3	**4,4**	4,5
5	**5,1**	5,2	5,3	5,4	**5,5**

principală: i = j secundară: i + j = n + 1 = 6

Diagonalele

- **diagonala principală:** `i == j` (de sus-stânga la jos-dreapta);
- **diagonala secundară:** `i + j == n + 1` (de sus-dreapta la jos-stânga), pentru indici de la 1. Dacă indicii pornesc de la 0, devine `i + j == n - 1`.

DE CE AȘA?

De ce `n + 1`? Pe diagonala secundară, când linia crește cu 1, coloana scade cu 1, deci suma `i + j` rămâne constantă. Primul element al ei este `(1, n)` și `1 + n = n + 1`. Verifică și pe ultimul: `(n, 1)` dă tot `n + 1`.

Pentru sumele celor două diagonale ajunge **o singură buclă**, cu `n` pași, nu două imbricate cu `n²`:

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> a[i][j];

    int sp = 0, ss = 0;
    for (int i = 1; i <= n; i++)
    {
        sp += a[i][i];                // diagonala principală
        ss += a[i][n - i + 1];        // coloana lui i pe diagonala secundară
    }
    cout << sp << ' ' << ss;
    return 0;
}
```

DE CE AȘA?

De ce `n - i + 1` ca indice de coloană? Din `i + j = n + 1` rezultă `j = n + 1 − i`. Alegem linia `i`, iar formula ne dă coloana pe care se află elementul diagonalei.

Urmărire (matricea `1 2 3 / 4 5 6 / 7 8 9`, `n = 3`):

i	`a[i][i]`	`a[i][n−i+1]`	sp	ss
(început)			0	0
1	`a[1][1] = 1`	`a[1][3] = 3`	1	3
2	`a[2][2] = 5`	`a[2][2] = 5`	6	8
3	`a[3][3] = 9`	`a[3][1] = 7`	15	15

Elementul din centru (la `n` impar) aparține **ambelor** diagonale și este numărat în ambele sume.

Intrare / ieșire

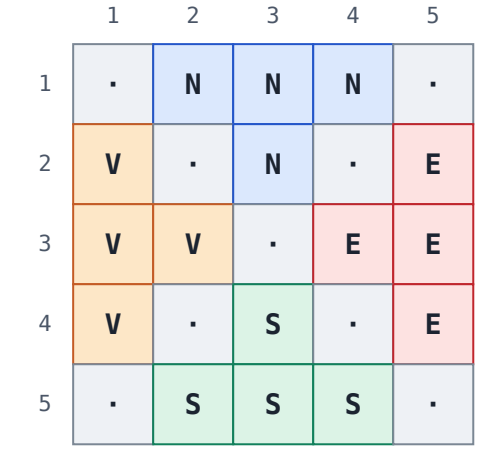
Intrare	Ieșire

`3`
`1 2 3`
`4 5 6`
`7 8 9`
	`15 15`

`2`
`1 2`
`3 4`
	`5 5`

2. Zonele determinate de diagonale

Cele două diagonale împart matricea în 4 zone: **Nord, Sud, Vest, Est**.



ı+1 S: i>j și i+j>n+1 V: i>j și i+j<n+1 E: i

Zonele

Zonă	Condiție (indici de la 1)
deasupra diag. principale	`i < j`
sub diag. principală	`i > j`
deasupra diag. secundare	`i + j < n + 1`
sub diag. secundare	`i + j > n + 1`
Nord	`i < j && i + j < n + 1`
Sud	`i > j && i + j > n + 1`
Vest	`i > j && i + j < n + 1`
Est	`i < j && i + j > n + 1`

DE CE AȘA?

De ce se combină două condiții? O zonă este „între" două diagonale: de partea unei diagonale **și** de partea celeilalte. De exemplu Nordul este deasupra principalei (`i < j`) și deasupra secundarei (`i + j < n + 1`).

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> a[i][j];

    int N = 0, S = 0, V = 0, E = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
        {
            if (i < j && i + j < n + 1) N += a[i][j];
            if (i > j && i + j > n + 1) S += a[i][j];
            if (i > j && i + j < n + 1) V += a[i][j];
            if (i < j && i + j > n + 1) E += a[i][j];
        }
    cout << N << ' ' << S << ' ' << V << ' ' << E;
    return 0;
}
```

La `n = 3`: Nord = `a[1][2] = 2`, Sud = `a[3][2] = 8`, Vest = `a[2][1] = 4`, Est = `a[2][3] = 6`; elementele de pe diagonale nu aparțin niciunei zone.

SFAT

La grile cu „care este condiția...?” desenează o matrice 4×4 sau 5×5 și încearcă formula pe 2–3 elemente alese de tine, unul din fiecare zonă.

3. Simetrie și transpusă

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

a

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

transpusa: t[i][j] = a[j][i]

Transpusa

- Matricea e **simetrică** față de diagonala principală dacă `a[i][j] == a[j][i]` pentru toți `i, j`.
- **Transpusa** are `t[i][j] = a[j][i]`: liniile devin coloane.

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> a[i][j];

    bool sim = true;
    for (int i = 1; i <= n; i++)
        for (int j = i + 1; j <= n; j++)      // doar deasupra diagonalei
            if (a[i][j] != a[j][i]) sim = false;
    cout << (sim ? "DA" : "NU");
    return 0;
}
```

DE CE AȘA?

De ce `j` pornește de la `i + 1`? Condiția `a[i][j] == a[j][i]` pentru perechea `(i, j)` este aceeași cu cea pentru `(j, i)`. Verificăm fiecare pereche o singură dată (`n(n−1)/2` perechi) și sărim diagonala, unde `a[i][i] == a[i][i]` e mereu adevărat.

4. Generări de matrice (Bac II.3)

Cerința tipică: „Scrieți un program care construiește în memorie tabloul cu proprietatea... și îl afișează.” Strategia:

1. scrie pe hârtie matricea din exemplu;
2. caută o **formulă** pentru `a[i][j]` în funcție de `i`, `j` (și `n`), eventual pe zone;
3. umple matricea cu două bucle și verifică formula pe 2–3 celule.

	1	2	3	4
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6
4	4	5	6	7

a[i][j] = i + j − 1

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	2	2	1
3	1	2	3	2	1
4	1	2	2	2	1
5	1	1	1	1	1

[i][j] = min(i, j, n−i+1, n−j+1) (chenare

Formule

Exemplul 1: o formulă simplă. Matricea din prima figură: `a[i][j] = i + j - 1`.

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            a[i][j] = i + j - 1;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++) cout << a[i][j] << ' ';
        cout << '\n';
    }
    return 0;
}
```

Exemplul 2: pe zone. Diagonala principală 0, deasupra ei 1, dedesubt 2:

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            if (i == j) a[i][j] = 0;
            else if (i < j) a[i][j] = 1;
            else a[i][j] = 2;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++) cout << a[i][j] << ' ';
        cout << '\n';
    }
    return 0;
}
```

Exemplul 3: chenare concentrice (marginea 1, apoi 2, ...): valoarea este **distanța până la cea mai apropiată margine**, deci

`a[i][j] = min(min(i, j), min(n − i + 1, n − j + 1))`.

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            a[i][j] = min(min(i, j), min(n - i + 1, n - j + 1));

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++) cout << a[i][j] << ' ';
        cout << '\n';
    }
    return 0;
}
```

DE CE AȘA?

De ce `min` din patru valori? `i - 1` este distanța până la marginea de sus, `n - i` până la cea de jos, `j - 1` și `n - j` până la stânga și dreapta. Chenarul în care se află celula este distanța minimă + 1, adică exact `min(i, j, n - i + 1, n - j + 1)`.

Exemplul 4: din vecini. Dacă prima linie și prima coloană sunt 1 și restul se calculează din celulele deja umplute, `a[i][j] = a[i-1][j] + a[i][j-1]`:

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            if (i == 1 || j == 1) a[i][j] = 1;
            else a[i][j] = a[i - 1][j] + a[i][j - 1];

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++) cout << a[i][j] << ' ';
        cout << '\n';
    }
    return 0;
}
```

DE CE AȘA?

De ce funcționează? Parcurgem liniile de sus în jos și coloanele de la stânga la dreapta, deci când ajungem la `(i, j)`, celulele `(i–1, j)` și `(i, j–1)` au fost deja calculate.

Urmărire (`a[3][3]` din exemplul 4): `a[3][3] = a[2][3] + a[3][2] = 3 + 3 = 6`.

5. Parcurgeri speciale: șerpuit și spirală

	1	2	3	4
1	1	2	3	4
2	12	13	14	5
3	11	16	15	6
4	10	9	8	7

spirală (chenar cu chenar)

	1	2	3	4
1	1	2	3	4
2	8	7	6	5
3	9	10	11	12
4	16	15	14	13

: impare de la stânga, cele pare c

Șerpuit și spirală

Șerpuit: liniile impare de la stânga la dreapta, cele pare invers.

```
for (int i = 1; i <= n; i++)
    if (i % 2 == 1) for (int j = 1; j <= n; j++) cout << a[i][j] << ' ';
    else            for (int j = n; j >= 1; j--) cout << a[i][j] << ' ';
```

Spirală: chenar cu chenar, în patru mișcări (sus →, dreapta ↓, jos ←, stânga ↑).

```cpp
#include <iostream>
using namespace std;

int a[101][101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> a[i][j];

    for (int k = 1; k <= (n + 1) / 2; k++)          // chenarul k; ultima linie/coloană a lui: n-k+1
    {
        int u = n - k + 1;
        for (int j = k; j <= u; j++) cout << a[k][j] << ' ';          // sus, spre dreapta
        for (int i = k + 1; i <= u; i++) cout << a[i][u] << ' ';      // dreapta, în jos
        if (u > k)
        {
            for (int j = u - 1; j >= k; j--) cout << a[u][j] << ' ';  // jos, spre stânga
            for (int i = u - 1; i > k; i--) cout << a[i][k] << ' ';   // stânga, în sus
        }
    }
    return 0;
}
```

DE CE AȘA?

De ce `if (u > k)`? Când `n` este impar, ultimul „chenar” este o singură celulă (centrul). Fără verificare, ultimele două mișcări ar afișa din nou elemente deja scrise. La `n = 3`, `k = 2`: `u = 2 = k`, deci doar celula `a[2][2]` este afișată, la prima mișcare.

6. Greșeli frecvente

Greșeala	Efect	Corect
`i + j == n` pentru diagonala secundară (indici de la 1)	diagonală deplasată	`i + j == n + 1`
două bucle imbricate pentru a aduna o diagonală	funcționează, dar face n^2 pași	o singură buclă
`j` de la 1 în loc de `i + 1` la verificarea simetriei	verifică de două ori; la `a[i][i]` nu strică, dar e inutil	`j = i + 1`
formula verificată doar pe un element	greșeală la margini	verifică pe 2–3 celule din zone diferite
afișarea spiralei fără `if (u > k)`	element centrul/dublat la `n` impar	tratează chenarul de lățime 1

7. La BAC

> **LA BAC**
>
> Subiectul II.3 cere des „construiește matricea…": descoperă formula, nu da valori „din mână". În Subiectul I apar grile cu zone (`i < j` , `i + j > n + 1`) și cu diagonale. Dacă enunțul numerotează liniile și coloanele de la **0**, diagonala secundară este `i + j == n - 1` . Citește mereu de unde pornesc indicii.

8. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Scrie condiția pentru elementele situate **pe sau deasupra** diagonalei secundare (indici de la 1).

2. Desenează matricea cu `a[i][j] = |i - j|` pentru `n = 4` .

3. Construiește matricea „tablă de șah" cu 0 și 1, cu `a[1][1] = 1` , pentru `n = 4` . Ce formulă folosești?

4. Câte elemente se află strict deasupra diagonalei secundare într-o matrice 20×20?

5. Pentru `a[i][j] = i + j - 1` , `n = 4` , calculează suma de pe diagonala principală și suma de pe cea secundară.

6. Pentru `n = 5` , în ce zonă (N, S, V, E sau pe diagonală) se află celulele `(2, 5)` , `(4, 1)` , `(3, 3)` ?

7. Desenează matricea cu `a[i][j] = min(i, j)` , pentru `n = 4` .

8. Ce afișează parcurgerea în spirală pentru matricea `1 2 3 / 4 5 6 / 7 8 9` ?

9. Care este transpusa matricei `1 2 3 / 4 5 6` (2 linii, 3 coloane)? Poate fi simetrică? De ce?

10. Câte perechi `(i, j)` trebuie comparate la verificarea simetriei unei matrice cu `n = 5` ?

11. În matricea din exemplul 4 (`n = 4`), cât este `a[3][4]` ?

9. Mai departe (pentru cine vrea mai mult)

Rotirea unei matrice cu 90°. Elementul `a[i][j]` ajunge pe poziția `(j, n - i + 1)` . Se obține facil prin transpunere urmată de oglindirea fiecărei linii. Folosit la procesarea imaginilor și în jocuri.

Triunghiul lui Pascal. Matricea din exemplul 4 este, citită pe diagonale, triunghiul lui Pascal (numerele de combinări). `a[i][j]` = C(i+j−2, i−1). Poți arăta singur că `a[3][4] = C(5, 2) = 10` .

Pătrate magice. Un pătrat în care toate liniile, coloanele și diagonalele au aceeași sumă. Pentru `n` impar există o construcție simplă (metoda siameză); caută-o ca exercițiu de generare.

Programare dinamică pe matrice. Exemplul 4 („valoarea vine din vecinii deja calculați") este ideea de bază a programării dinamice, pe care o vei folosi la clasa a XI-a (de exemplu: câte drumuri sunt de la colțul din stânga-sus la cel din dreapta-jos).

Rezumat

- Principală: `i == j` ; secundară: `i + j == n + 1` (indici de la 1).

- Zonele: combinații de `i < j`, `i > j`, `i + j < n + 1`, `i + j > n + 1`.
- Simetrie: compari `a[i][j]` cu `a[j][i]` pentru `j > i`; transpusa: `t[i][j] = a[j][i]`.
- Generări: găsește formula pentru `a[i][j]`, verific-o pe exemplu; uneori valoarea se calculează din vecinii deja umpluți.
- Șerpuit: inversezi sensul pe liniile pare; spirală: chenar cu chenar.

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] `#313` **diagonale** — ușoară
  https://www.pbinfo.ro/probleme/313/diagonale
- [ ] `#729` **zona** — medie
  https://www.pbinfo.ro/probleme/729/zona
- [ ] `#206` **genmat1** — ușoară
  https://www.pbinfo.ro/probleme/206/genmat1
- [ ] `#207` **genmat2** — ușoară
  https://www.pbinfo.ro/probleme/207/genmat2

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] `#786` **matsim** — ușoară
  https://www.pbinfo.ro/probleme/786/matsim
- [ ] `#780` **cmmdcsum** — ușoară
  https://www.pbinfo.ro/probleme/780/cmmdcsum
- [ ] `#783` **diagonale1** — medie
  https://www.pbinfo.ro/probleme/783/diagonale1
- [ ] `#781` **zone1** — medie
  https://www.pbinfo.ro/probleme/781/zone1
- [ ] `#213` **genmat4** — ușoară
  https://www.pbinfo.ro/probleme/213/genmat4
- [ ] `#214` **genmat5** — ușoară
  https://www.pbinfo.ro/probleme/214/genmat5
- [ ] `#216` **genmat7** — ușoară
  https://www.pbinfo.ro/probleme/216/genmat7

- [ ] #210 **chenar** — ușoară
  https://www.pbinfo.ro/probleme/210/chenar

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #211 **spirala** — medie
  https://www.pbinfo.ro/probleme/211/spirala
- [ ] #794 **serpuire** — medie
  https://www.pbinfo.ro/probleme/794/serpuire
- [ ] #286 **chenare** — medie
  https://www.pbinfo.ro/probleme/286/chenare

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #147 **Exercițiul #147**
  tablou n×n: care element se află pe o anumită poziție față de diagonale
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=147
- [ ] #212 **Exercițiul #212**
  câte elemente sunt strict deasupra diagonalei secundare (20×20)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=212
- [ ] #215 **Exercițiul #215**
  matrice 5×5 cu numerele 1..25: elementele de pe diagonale
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=215
- [ ] #203 **Exercițiul #203**
  matrice 5×5: valoare calculată cu o secvență de program
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=203
- [ ] #1249 **Exercițiul #1249**
  generare: ce secvență construiește o matrice 4×4 dată
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1249
- [ ] #1255 **Exercițiul #1255**
  generare: ce secvență construiește o matrice 5×5 dată
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1255
- [ ] #1274 **Exercițiul #1274**
  generare: ce secvență construiește o matrice 4×4 dată
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1274

Citește și exersează în plus

- Tablouri pătratice https://www.pbinfo.ro/articole/5626/tablouri-patratice
- Wikipedia — Matrice (transpusa, matrice simetrică) — definițiile matematice din spatele transpusei și ale matricelor simetrice https://ro.wikipedia.org/wiki/Matrice_(matematică)
- Python Tutor pentru C++ — urmărește cum se umple matricea pas cu pas https://pythontutor.com/cpp.html
- Triunghiul lui Pascal — matricea din secțiunea 5 este un triunghi al lui Pascal „întors" https://ro.wikipedia.org/wiki/Triunghiul_lui_Pascal

Răspunsuri

Exersează pe hârtie

1. `i + j <= n + 1`.

2.

```
0 1 2 3
1 0 1 2
2 1 0 1
3 2 1 0
```

3. `a[i][j] = (i + j) % 2 == 0 ? 1 : 0` (la `i = j = 1` suma 2 este pară, deci 1):

```
1 0 1 0
0 1 0 1
1 0 1 0
0 1 0 1
```

4. Condiția este `i + j < 21`. Pentru linia `i` există `20 − i` coloane, deci 19 + 18 + … + 1 = **190**.

5. Principala: `a[i][i] = 2i − 1`, deci 1 + 3 + 5 + 7 = **16**. Secundara: `i + j = 5`, deci `a[i][j] = 4` pe fiecare din cele 4 poziții: **16**.

6. `(2, 5)`: `i < j` și `i + j = 7 > 6` → **Est**. `(4, 1)`: `i > j` și `i + j = 5 < 6` → **Vest**. `(3, 3)`: `i = j` (și `i + j = 6`) → pe **ambele diagonale**.

7.

```
1 1 1 1
1 2 2 2
1 2 3 3
1 2 3 4
```

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

9. Transpusa are 3 linii și 2 coloane: `1 4 / 2 5 / 3 6`. O matrice simetrică trebuie să fie pătratică (altfel `a[i][j]` și `a[j][i]` nu există amândouă), deci aceasta nu poate fi simetrică.

10. Perechile cu `i < j`: 5 · 4 / 2 = **10**.

11. `a[3][4] = a[2][4] + a[3][3] = 4 + 6 = 10`.