INFORMATICĂ · CLASA A X-A · TABLOURI BIDIMENSIONALE · LECȚIA 8

Tablouri bidimensionale (matrice): declarare, parcurgere pe linii și coloane

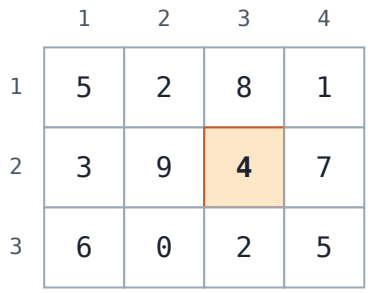
Durată: 60 min Nivel: toate profilurile Înainte: lecțiile x-02

CE VEI ȘTI DUPĂ LECȚIE

- Declari, citești și afișezi o matrice cu n linii și m coloane.
- Parcurgi matricea pe linii și pe coloane.
- Calculezi sume / maxime pe fiecare linie sau coloană.
- Interschimbi linii sau coloane și elimini o linie / o coloană.

1. Ideea: un tabel în memorie

Până acum un vector era o **listă**. Dar multe date sunt un **tabel**: notele a 3 elevi la 4 teme, o hartă de joc, o tablă de șah. Un astfel de tabel se numește **matrice** (tablou bidimensional): are **linii** și **coloane**, iar un element se află cu două numere, ca o celulă în Excel: linia `i`, coloana `j`.



a[2][3] = 4 (linia 2, coloana 3)

Matrice

Aici `n = 3` linii, `m = 4` coloane și `a[2][3] = 4`.

DE CE AȘA?

De ce două indici? O poziție dintr-un tabel nu se poate descrie cu un singur număr fără o regulă de numerotare. Două numere (linia, coloana) sunt naturale și corespund exact celor două bucle imbricate cu care vom parcurge matricea: o buclă pentru linii, una pentru coloane.

2. Declarare, citire, afișare

```
int a[101][101];   // 101 linii × 101 coloane; folosim indicii 1..100
```

DE CE AȘA?

De ce `101` și nu `100`? Dacă vrem să numerotăm de la 1 până la 100, avem nevoie de poziția 100, deci dimensiunea trebuie să fie cu 1 mai mare (poziția 0 rămâne nefolosită). **De ce global?** O matrice mare pusă în `main` poate umple stiva programului; în plus, variabilele globale pornesc cu 0, ceea ce ne scutește de inițializare.

Programul citește `n`, `m` și apoi elementele, rând cu rând, și le afișează la fel:

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

int a[101][101], n, m;

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

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
            cout << a[i][j] << ' ';
        cout << '\n';                  // rând nou după fiecare LINIE a matricei
    }
    return 0;
}
```

DE CE AȘA?

De ce `cout << '\n'` este după bucla cu `j`, nu în ea? Bucla cu `j` scrie o linie întreagă. Dacă `'\n'` ar fi în ea, fiecare număr ar ajunge pe rând separat și matricea ar arăta ca o coloană.

Intrare / ieșire

Intrare	Ieșire

`2 3`
`1 2 3`
`4 5 6`
	`1 2 3`
	`4 5 6`

`1 4` `7 7 0 9`	`7 7 0 9`

Cum se execută citirea (pentru `n = 2`, `m = 2`, valorile `8 5 3 1`):

i	j	se citește	efect
1	1	8	`a[1][1] = 8`
1	2	5	`a[1][2] = 5`
2	1	3	`a[2][1] = 3`

i	j	se citește	efect
2	2	1	`a[2][2] = 1`

Observă: `j` se schimbă cel mai repede, `i` doar după ce `j` a parcurs toată linia.

3. Parcurgerea pe linii și pe coloane

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12

pe linii: i exterior, j interior

	1	2	3	4
1	1	4	7	10
2	2	5	8	11
3	3	6	9	12

pe coloane: j exterior, i interior

Ordinea de parcurgere

Regula de aur: **ce vrei „pentru fiecare linie" → bucla pe `i` e cea exterioară; ce vrei „pentru fiecare coloană" → bucla pe `j` e cea exterioară.**

Programul de mai jos afișează suma fiecărei linii, apoi suma fiecărei coloane:

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

int a[101][101], n, m;

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

    for (int i = 1; i <= n; i++)          // pentru fiecare LINIE
    {
        int s = 0;                        // se reinițializează la fiecare linie
        for (int j = 1; j <= m; j++)
            s += a[i][j];
        cout << s << ' ';
    }
    cout << '\n';

    for (int j = 1; j <= m; j++)          // pentru fiecare COLOANĂ
    {
        int s = 0;
        for (int i = 1; i <= n; i++)
            s += a[i][j];
        cout << s << ' ';
    }
    return 0;
}
```

Urmărirea pentru prima linie a matricei din figură (`5 2 8 1`):

j	a[1][j]	s
(început)		0
1	5	5
2	2	7
3	8	15
4	1	16

Se afișează 16; apoi `s` se pune din nou pe 0 și se continuă cu linia 2 (23) și linia 3 (13).

DE CE AȘA?

De ce `int s = 0;` este între cele două bucle? Suma trebuie să pornească de la 0 **pentru fiecare linie în parte**. Dacă `s = 0` ar fi înaintea ambelor bucle, suma s-ar acumula peste linii și ai obține 16, 39, 52 în loc de 16, 23, 13.

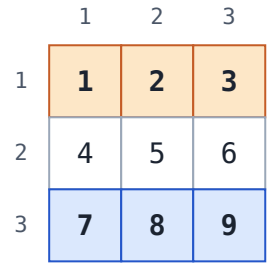
Intrare / ieșire

Intrare	Ieșire

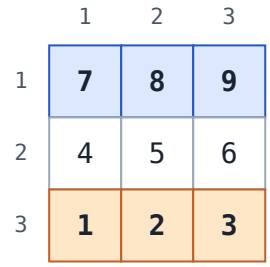
`3 4`
`5 2 8 1`
`3 9 4 7`
`6 0 2 5`
	`16 23 13`
	`14 11 14 13`

`2 2`
`1 1`
`1 1`
	`2 2`
	`2 2`

4. Interschimbarea a două linii (sau coloane)



înainte



vap(a[1][j], a[3][j]) pentru f

Interschimbare

Pentru liniile `p` și `q` interschimbăm, coloană cu coloană, elementele aflate pe aceeași coloană `j`:

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

int a[101][101], n, m;

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

    for (int j = 1; j <= m; j++)          // O SINGURĂ buclă: linia are m elemente
        swap(a[p][j], a[q][j]);

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

DE CE AȘA?

De ce o singură buclă? Interschimbăm doar două linii, nu toată matricea. Fiecare pas schimbă o pereche de elemente (a[p][j] cu a[q][j]), deci ne trebuie `m` pași. Pentru **coloane** bucla ar merge pe `i` și am scrie `swap(a[i][p], a[i][q])`.

5. Eliminarea unei linii

Ca la vectori: mutăm liniile de dedesubt cu un loc mai sus, apoi micșorăm `n`.

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

int a[101][101], n, m;

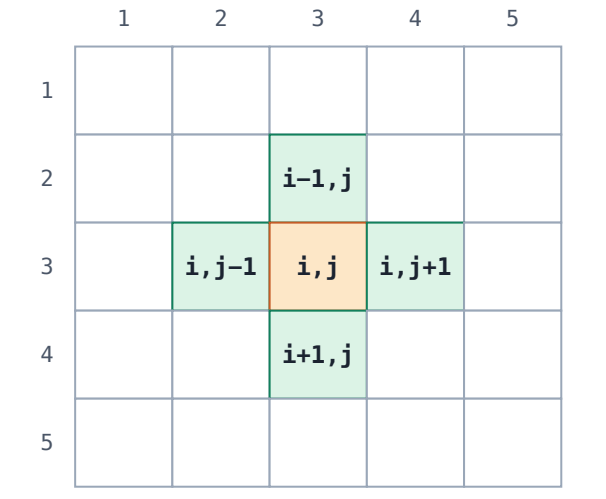
int main()
{
    int k;
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            cin >> a[i][j];
    cin >> k;

    for (int i = k; i < n; i++)           // linia i primește conținutul liniei i+1
        for (int j = 1; j <= m; j++)
            a[i][j] = a[i + 1][j];
    n--;                                  // matricea are acum o linie mai puțin

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

Pentru `k = 2` se întâmplă: linia 2 ← linia 3; apoi `n` devine 2. Linia 3 veche mai există în memorie, dar nu mai este parcursă.

6. Vecinii unui element



sus, jos, stânga, dreapta; la margine unii nu există

Vecinii

Multe probleme (jocuri, hărți, „eroziune”) compară un element cu cei 4 vecini. În loc să scriem 4 `if`-uri aproape identice, folosim doi vectori de **direcții**:

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

int a[101][101], n, m;
int di[] = {-1, 0, 1, 0};          // schimbarea liniei: sus, dreapta, jos, stânga
int dj[] = {0, 1, 0, -1};          // schimbarea coloanei

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

    int cnt = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
            bool varf = true;
            for (int d = 0; d < 4; d++)
            {
                int ni = i + di[d], nj = j + dj[d];
                if (ni >= 1 && ni <= n && nj >= 1 && nj <= m)   // vecinul există?
                    if (a[ni][nj] >= a[i][j]) varf = false;
            }
            if (varf) cnt++;
        }
    cout << cnt;               // câte elemente sunt strict mai mari decât toți vecinii
    return 0;
}
```

DE CE AȘA?

De ce verificăm `ni >= 1 && ni <= n …`? Un element de pe margine nu are 4 vecini. Dacă am citi `a[0][j]` sau `a[n+1][j]`, am obține valori care nu fac parte din matrice (în cel mai bun caz 0, în cel mai rău ieșim din zona de memorie). Verificarea apare **înainte** de citirea lui `a[ni][nj]`.

Urmărire pentru prima intrare (`1 2 3 / 4 9 6 / 7 8 5`), doar câteva celule:

(i, j)	a[i][j]	vecinii existenți	un vecin ≥ ?	varf
(1, 1)	1	2, 4	da (2)	nu
(2, 2)	9	2, 6, 8, 4	nu	**da**
(1, 3)	3	2, 6	da (6)	nu
(3, 2)	8	9, 7, 5	da (9)	nu

Doar elementul 9 este „vârf”, deci se afișează 1.

7. Greșeli frecvente

Greșeala	Efect	Corect
`a[i, j]`	virgula e operator: se folosește doar `j`	`a[i][j]`
liniile și coloanele inversate în buclă (`j <= n`)	element în afara matricei sau parcurgere incompletă	linii: `i` până la `n` ; coloane: `j` până la `m`
`int s = 0;` înaintea ambelor bucle	sumele se acumulează peste linii	`s = 0` pentru fiecare linie
acces la vecin fără verificarea marginilor	valori aiurea / crash	verifică `1 ≤ ni ≤ n` , `1 ≤ nj ≤ m`
`int a[100][100]` și folosirea indicelui 100	ieșire din tablou	dimensiune cu 1 mai mare

8. La BAC

LA BAC

Subiectul I conține aproape mereu o secvență cu matrice (ce se afișează, câte elemente verifică o condiție), iar Subiectul II.3 sau III cere construirea sau prelucrarea unei matrice. Ține minte: **numărul de linii, numărul de coloane și de unde pornesc indicii (0 sau 1)** se citesc cu atenție din enunț. La grile, scrie matricea pe hârtie și simulează 2–3 pași.

9. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

Matricea de lucru: `5 2 8 1 / 3 9 4 7 / 6 0 2 5` (cea din figură).

1. Calculează suma elementelor de pe coloana 3 și maximul liniei 2.

2. Ce afișează?

```
for (int j = 1; j <= 4; j++)
{
    int s = 0;
    for (int i = 1; i <= 3; i++) s += a[i][j];
    cout << s << ' ';
}
```

3. Care sunt cei 4 vecini ai lui `a[2][3]` (sus, dreapta, jos, stânga)?

4. Ce matrice obții dacă interschimbi liniile 1 și 3?

5. Ce matrice obții dacă elimini linia 2 (`k = 2`)? Cât este `n` după eliminare?

6. Pentru matricea `1 2 3 / 4 5 6 / 7 8 9` , ce afișează?

```cpp
for (int i = 1; i <= 2; i++)
    for (int j = 1; j <= 3; j++)
        cout << a[j][i] << ' ';
```

7. Ce afișează acest cod pentru matricea de lucru? Care e greșeala față de „suma fiecărei linii”?

```cpp
int s = 0;
for (int i = 1; i <= 3; i++)
{
    for (int j = 1; j <= 4; j++) s += a[i][j];
    cout << s << ' ';
}
```

8. Câte elemente are `int a[4][5]` și care este ultimul element valid?

9. (BAC) O matrice are 20 de linii și 20 de coloane, numerotate de la 1 la 20. Ce valoare are `a[i][j]` pentru `i = 20`, `j = 20`, dacă toate elementele au fost inițializate cu `a[i][j] = i - j`?

10. Mai departe (pentru cine vrea mai mult)

Matrice și hărți. Un joc 2D (labirint, Minesweeper, Game of Life) este o matrice, iar mutarea personajului este exact schimbarea `(i, j) → (i + di[d], j + dj[d])`. Cu un mic algoritm numit **fill** (sau BFS pe matrice) afli, de exemplu, câte „insule” sunt într-o hartă: o să-l întâlnești la clasa a XI-a.

Memorie. Matricea este, de fapt, un vector de vectori așezat linie după linie în memorie. De aceea parcurgerea **pe linii** este mai rapidă decât pe coloane la matrice foarte mari (memoria cache): la 5000×5000, diferența se vede.

Vector de vectori. În C++ modern: `vector<vector<int>> a(n + 1, vector<int>(m + 1));` are dimensiunea stabilită la execuție, fără limite fixe.

Imagini. O poză alb-negru este o matrice de luminozități. Un filtru „blur” înlocuiește fiecare pixel cu media vecinilor lui: exact tiparul din secțiunea 6.

Rezumat

- `a[i][j]`: linia `i`, coloana `j`; `n` linii, `m` coloane.
- Citire/afișare/prelucrare = două bucle imbricate; ce vrei „pentru fiecare” merge în bucla exterioară, iar inițializările între bucle.
- Interschimbare de linii: o buclă pe coloane; eliminare: mută liniile de dedesubt.
- Vecini: vectorii `di`, `dj` + verificarea marginilor.

Probleme de lucru (pbinfo)

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

- [ ] **#658** **sumalinii** — ușoară
  https://www.pbinfo.ro/probleme/658/sumalinii

- [ ] #662 **sumelpare** — ușoară
  https://www.pbinfo.ro/probleme/662/sumelpare
- [ ] #191 **maxmat** — ușoară
  Maximul fiecărei linii și coloana lui.
  https://www.pbinfo.ro/probleme/191/maxmat
- [ ] #619 **sortlin** — ușoară
  https://www.pbinfo.ro/probleme/619/sortlin

Temă (pbinfo)

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

- [ ] #666 **nrprime** — ușoară
  https://www.pbinfo.ro/probleme/666/nrprime
- [ ] #192 **sumcolmax** — ușoară
  https://www.pbinfo.ro/probleme/192/sumcolmax
- [ ] #222 **mincols** — ușoară
  https://www.pbinfo.ro/probleme/222/mincols
- [ ] #316 **mincols2** — ușoară
  https://www.pbinfo.ro/probleme/316/mincols2
- [ ] #766 **nrpare** — ușoară
  https://www.pbinfo.ro/probleme/766/nrpare
- [ ] #804 **colegale** — ușoară
  https://www.pbinfo.ro/probleme/804/colegale
- [ ] #193 **interschimbarelinii** — ușoară
  https://www.pbinfo.ro/probleme/193/interschimbarelinii
- [ ] #3120 **simetrica-bac** — ușoară
  Subiect de Bac.
  https://www.pbinfo.ro/probleme/3120/simetrica-bac

Pentru cei care vor mai mult

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

- [ ] #796 **eliminare3** — ușoară
  https://www.pbinfo.ro/probleme/796/eliminare3
- [ ] #4183 **bac-eroziune** — ușoară
  https://www.pbinfo.ro/probleme/4183/bac-eroziune

Grile din subiectele de BAC (pbinfo)

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

- [ ] #70 **Exercițiul #70**
  care expresie este un element valid al tabloului int a[5][6]
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=70

- [ ] #74 **Exercițiul #74**
  declararea corectă a unui tablou bidimensional cu 20 de elemente
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=74

- [ ] #121 **Exercițiul #121**
  parcurgerea unei matrice 20×20 și ce se obține
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=121

- [ ] #156 **Exercițiul #156**
  secvență care umple o matrice după o regulă; ce valoare ajunge în a[i][j]
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=156

- [ ] #194 **Exercițiul #194**
  matrice 5×5: valoarea calculată cu o buclă dublă
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=194

- [ ] #260 **Exercițiul #260**
  valori afișate după interschimbări într-o matrice
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=260

Citește și exersează în plus

- Tablouri bidimensionale https://www.pbinfo.ro/articole/5620/tablouri-bidimensionale
- Declararea matricelor https://www.pbinfo.ro/articole/5622/declararea-matricelor-referirea-elementelor
- Parcurgerea matricelor https://www.pbinfo.ro/articole/5623/parcurgerea-tablourilor-bidimensionale
- Python Tutor pentru C++ (vezi matricea în memorie) — scrie un program cu int a[3][4] și urmărește pas cu pas bucla dublă https://pythontutor.com/cpp.html
- CSES — Grid Paths / Counting Rooms (probleme cu matrice) — pentru cine vrea mai mult: hărți pe matrice, cu vecini (engleză) https://cses.fi/problemset/
- cppreference — array declaration — referință oficială pentru tablouri multidimensionale (engleză) https://en.cppreference.com/w/cpp/language/array

Răspunsuri

Exersează pe hârtie

1. Coloana 3: 8 + 4 + 2 = **14**. Maximul liniei 2 (`3 9 4 7`): **9**.

2. `14 11 14 13` (sumele coloanelor: 5+3+6, 2+9+0, 8+4+2, 1+7+5).

3. Sus `a[1][3] = 8`, dreapta `a[2][4] = 7`, jos `a[3][3] = 2`, stânga `a[2][2] = 9`.

4. `6 0 2 5 / 3 9 4 7 / 5 2 8 1`.

5. `5 2 8 1 / 6 0 2 5`, iar `n = 2`.

6. `1 4 7 2 5 8`: pentru `i = 1` se afișează coloana 1 (`a[1][1], a[2][1], a[3][1]`), pentru `i = 2` coloana 2. Indicii sunt „răsturnați", deci parcurgerea e pe coloane.

7. Se afișează `16 39 52`. Greșeala: `s = 0` e înaintea ambelor bucle, deci sumele se adună peste linii. Corect: `s = 0` la începutul fiecărei linii, ca să obții `16 23 13`.

8. 4 · 5 = **20** de elemente; indicii valizi sunt 0..3 și 0..4, deci ultimul este `a[3][4]`.

9. `a[20][20] = 20 − 20 = 0`.