FIȘĂ DE LUCRU · CLASA A X-A · FIȘA 7 · ACOPERĂ LECȚIILE X.8–X.9

Matrice

Durată: 120 min Nivel: toate profilurile

CE VEI ȘTI DUPĂ LECȚIE

- Parcurgi o matrice pe linii, pe coloane, pe diagonale, pe zone și printre vecini, fără să ieși din ea.
- Construiești o matrice după o regulă (Bac II.3) și găsești formula unei matrice date.
- Transformi o matrice: transpusă, oglindire, rotire, ștergere de coloană, sortare de linii.

Ce trebuie să știi (o pagină)

Ce vrei	Cod / condiție (indici de la 1)	Atenție
pentru fiecare linie / coloană	linii: `i` afară, `j` înăuntru; coloane: invers	`s = 0` **între** bucle; `i ≤ n`, `j ≤ m`
diagonala principală / secundară	`a[i][i]` / `a[i][n + 1 - i]`, o singură buclă	de la 0: `a[i][n - 1 - i]`
deasupra / sub principală, secundară	`i < j` / `i > j`; `i + j < n + 1` / `i + j > n + 1`	
zonele N, S, V, E	N: `i < j && i + j < n + 1`, S: `i > j && i + j > n + 1`, V: `i > j && i + j < n + 1`, E: `i < j && i + j > n + 1`	
4 vecini	`di[] = {-1, 0, 1, 0}, dj[] = {0, 1, 0, -1}`	verifici `1 ≤ x ≤ n`, `1 ≤ y ≤ m`
transpusa pe loc	`for (j = i + 1; j <= n; j++) swap(a[i][j], a[j][i]);`	`j` de la `i + 1`!
generare (Bac II.3)	formula `a[i][j] = f(i, j, n)` sau regula din vecini deja calculați	verifici pe 3 celule din zone diferite

SFAT

Înainte să scrii codul unei generări, scrie pe hârtie matricea pentru `n = 4` și, lângă fiecare celulă, perechea `(i, j)`. Formula apare de obicei din `i + j`, `i − j`, `min`, `max` sau din paritatea liniei.

Nivelul 1 · Prezice

★ citește și prezice

P1. Matricea `b` are 3 linii și 4 coloane. Ce afișează fiecare secvență?

i j	1	2	3	4
1	4	7	1	0
2	2	5	9	3
3	8	6	2	4

```
for (int j = 1; j <= 4; j++) {                    // a)
    int mn = b[1][j];
    for (int i = 2; i <= 3; i++) if (b[i][j] < mn) mn = b[i][j];
    cout << mn << ' ';
}
for (int i = 1; i <= 3; i++) {                    // b)
    int c = 0;
    for (int j = 1; j <= 4; j++) if (b[i][j] > b[i][1]) c++;
    cout << c << ' ';
}
int s = 0;                                        // c)
for (int i = 1; i <= 3; i++) s += b[i][4 - i + 1];
cout << s;
```

P2. Desenează matricea obținută cu `a[i][j] = i * j % 5`, pentru `n = 4`. Cât este suma diagonalei principale? Ce observi pe fiecare linie și pe fiecare coloană?

P3. Desenează matricea pentru `n = 3`. Apoi modifică în minte codul: coloanele pare se umplu **de jos în sus**. Desenează matricea nouă pentru `n = 4`.

```
int k = 0;
for (int j = 1; j <= n; j++)
    for (int i = 1; i <= n; i++)
        a[i][j] = ++k;
```

P4. Fără să desenezi toată matricea: pentru `n = 6`, câte elemente are fiecare zonă (N, S, E, V) și câte sunt pe diagonale (pe cel puțin una)? Dar pentru `n = 5`? Verifică: totalul trebuie să fie n^2.

P5. Matricea `1 2 3 / 4 5 6 / 7 8 9`. Desenează-o după primul pas, apoi după al doilea. Ce transformare face, în total?

```
for (int i = 1; i <= n; i++)                // pasul 1
    for (int j = i + 1; j <= n; j++) swap(a[i][j], a[j][i]);
for (int i = 1; i <= n; i++)                // pasul 2
    for (int j = 1; j <= n / 2; j++) swap(a[i][j], a[i][n - j + 1]);
```

P6. Minesweeper. Hartă de 3 × 4, `*` = mină. În fiecare celulă fără mină scrie câte mine are printre cei **8** vecini.

i j	1	2	3	4
1	*			*
2				
3		*		

P7. BAC, indici de la 0. Tabloul `b` are 5 linii și 5 coloane, numerotate de la 0 la 4, cu `b[i][j] = 5 * i + j`. Ce se afișează? Explică rezultatul fără să aduni toate elementele.

```
int s = 0;
for (int i = 0; i < 5; i++) s += b[i][4 - i] - b[i][i];
cout << s;
```

P8. Pentru `n = 4`, elementul de pe linia 3 și coloana 2 se află: deasupra sau sub diagonala principală? Deasupra sau sub cea secundară? În ce zonă? Dar celula `(1, 4)`? Ce se schimbă la `(3, 2)` dacă `n = 5`?

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Elevul vrea să transpună pe loc matricea `1 2 3 / 4 5 6 / 7 8 9`. Ce obține? De ce? Repară schimbând un singur lucru.

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        swap(a[i][j], a[j][i]);
```

P10. Matricea globală are `n = 2` linii, `m = 4` coloane: `3 1 4 1 / 5 9 2 6`. Programul trebuie să afișeze suma fiecărei coloane, adică `8 10 6 7`. Ce afișează? Găsește cele două greșeli.

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

P11. *Puzzle cu rânduri amestecate.* Șterge coloana `k` dintr-o matrice `n × m`. Pune rândurile în ordine; un rând este în plus. Test: `1 2 3 4 / 5 6 7 8 / 9 10 11 12`, `k = 2` → `1 3 4 / 5 7 8 / 9 11 12`.

```
A)  m--;
B)      for (int j = k; j < m; j++)
C)  for (int i = 1; i <= n; i++)
D)          a[i][j] = a[i][j + 1];
E)          a[i][j + 1] = a[i][j];
```

P12. Completează golurile ca programul să construiască matricea de mai jos (`n = 4`): pe diagonala secundară `0`, deasupra ei numărul liniei, sub ea numărul coloanei. Ce afișează pentru `n = 5`?

i j	1	2	3	4
1	1	1	1	0
2	2	2	0	4
3	3	0	3	4
4	0	2	3	4

```
if (__________) a[i][j] = 0;
else if (__________) a[i][j] = ___;
else a[i][j] = ___;
```

P13. „Vârfurile" unei hărți sunt celulele **strict mai mari** decât toți vecinii lor (sus, jos, stânga, dreapta) din hartă. Codul uită să verifice marginile; matricea e globală (în jur sunt zerouri). Pe harta `-5 -2 -7 / -1 -3 -4 / -6 -8 -9` afișează `0`. Care sunt vârfurile corecte? De ce a funcționat codul pe hărțile cu valori pozitive? Repară.

```
for (int d = 0; d < 4; d++)
    if (a[i + di[d]][j + dj[d]] >= a[i][j]) ok = false;
```

P14. Ce greșeală a făcut elevul? Matricea are 4 linii și 4 coloane, numerotate **de la 0**, cu `b[i][j] = 10 * (i + 1) + (j + 1)`. Ca să adune diagonala secundară, el testează `i + j == n + 1`. Ce sumă obține? Ce sumă trebuia?

Nivelul 3 · Scrie cod

★★ scrie cod

Fiecare matrice se citește după dimensiuni, linie cu linie. Verifică **toate** testele.

P15. Cinematograful. Sala are `n` rânduri și `m` locuri pe rând (`1` = ocupat, `0` = liber). Afișează rândul cu cele mai multe locuri libere (primul, la egalitate) și câte are.

intrare	ieșire
`3 5` / `1 0 0 1 1` / `0 0 1 0 0` / `1 1 1 1 1`	`2 4`
`2 2` / `1 1` / `1 1`	`1 0`
`1 3` / `0 0 0`	`1 3`

P16. Turnul de șah. Tabla `n × n` are piese (`1`) și locuri goale (`0`). Un turn stă pe `(x, y)` (loc gol). Câte piese atacă? Turnul vede pe linie și pe coloană, în cele 4 direcții, până la **prima** piesă întâlnită. Folosește vectorii de direcții.

intrare	ieșire
`4` / `0 1 0 0` / `1 0 0 1` / `0 0 0 0` / `0 1 1 0` / `2 2`	`4`
aceeași tablă, `3 3`	`1`
`1` / `0` / `1 1`	`0`

P17. Careul de Sudoku. Construiește matricea `n × n` cu prima linie `1 2 … n`, iar fiecare linie următoare este cea de deasupra rotită cu o poziție la stânga. Găsește o formulă pentru `a[i][j]`.

intrare	ieșire
`4`	`1 2 3 4` / `2 3 4 1` / `3 4 1 2` / `4 1 2 3`
`3`	`1 2 3` / `2 3 1` / `3 1 2`
`1`	`1`

P18. Oglinda pe secundară. Afișează `DA` dacă matricea pătratică este simetrică față de **diagonala secundară**, altfel `NU`. Ce element îi corespunde lui `a[i][j]`? Compară fiecare pereche o singură dată.

intrare	ieșire
`3` / `1 2 3` / `4 5 2` / `6 4 1`	`DA`
`2` / `1 2` / `2 3`	`NU`
`1` / `7`	`DA`

P19. Clasamentul pe linii. Sortează liniile matricei crescător după suma lor (la sume egale, liniile rămân în ordinea inițială). Calculează sumele o singură dată.

intrare	ieșire
`3 3` / `5 5 5` / `1 1 1` / `2 9 0`	`1 1 1` / `2 9 0` / `5 5 5`
`2 2` / `3 1` / `2 2`	`3 1` / `2 2`
`1 3` / `7 8 9`	`7 8 9`

P20. Jocul vieții. Fiecare celulă e vie (`1`) sau moartă (`0`). La pasul următor, o celulă vie cu 2 sau 3 vecini vii (din cei 8) rămâne vie, una moartă cu exact 3 vecini vii devine vie, iar restul sunt moarte. Afișează harta după **un** pas. Atenție: toate celulele se schimbă **simultan**.

intrare	ieșire
`5 5` / `0 0 0 0 0` / `0 0 0 0 0` / `0 1 1 1 0` / `0 0 0 0 0` / `0 0 0 0 0`	`0 0 0 0 0` / `0 0 1 0 0` / `0 0 1 0 0` / `0 0 1 0 0` / `0 0 0 0 0`
`4 4` / `0 0 0 0` / `0 1 1 0` / `0 1 1 0` / `0 0 0 0`	aceeași hartă
`3 3` / `1 1 1` / `1 1 1` / `1 1 1`	`1 0 1` / `0 0 0` / `1 0 1`

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Găsește formula. Matricea de mai jos (`n` linii, `m = 4` coloane) a fost umplută „șerpuit”. Scrie o formulă pentru `a[i][j]` **fără contor** (separat pentru liniile impare și pare). Cât este `a[100][3]` dacă `m = 10`? Dar `a[7][3]`?

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

P22. Fără `if`. Calculează suma zonei **Nord** parcurgând **doar** elementele ei, fără nicio condiție: de unde până unde merge `j` pe linia `i`? Câte iterații face bucla pentru `n = 6` și pentru `n = 7`? Compară cu `n²`.

P23. De ce funcționează rotirea? La P5, unde ajunge elementul de pe poziția `(i, j)` după cei doi pași? Scrie regula. Ce obții dacă aplici transformarea de două ori? Scrie direct formula pentru rotirea cu 180°.

P24. Matricea pusă pe un singur rând. Elementele unei matrice cu `n` linii și `m` coloane se scriu linie după linie într-un vector `v[1..n·m]`. Pe ce poziție `k` ajunge `a[i][j]`? Problema inversă: pentru un `k` dat, care sunt `i` și `j`, **fără buclă**? Verifică pe `n = 3, m = 4` cu `k = 8`, `k = 9`, `k = 12`.

P25. Mini-proiect: drumurile robotului. Un robot pleacă din `(1, 1)` și ajunge în `(4, 4)` mergând doar la **dreapta** sau **în jos**. Celulele `(2, 2)` și `(3, 4)` sunt blocate. Câte drumuri are? Ideea: în celula `(i, j)` se poate ajunge doar din `(i − 1, j)` sau din `(i, j − 1)`. Completează matricea cu numărul de drumuri până la fiecare celulă. Câte drumuri ar fi fără obstacole?

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] `#658` **sumalinii** — ușoară
  https://www.pbinfo.ro/probleme/658/sumalinii
- [ ] `#191` **maxmat** — ușoară
  https://www.pbinfo.ro/probleme/191/maxmat
- [ ] `#313` **diagonale** — ușoară
  https://www.pbinfo.ro/probleme/313/diagonale
- [ ] `#729` **zona** — medie
  https://www.pbinfo.ro/probleme/729/zona

Temă pentru acasă (pbinfo)

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

- [ ] `#193` **interschimbarelinii** — ușoară
  https://www.pbinfo.ro/probleme/193/interschimbarelinii
- [ ] `#3120` **simetrica-bac** — ușoară
  https://www.pbinfo.ro/probleme/3120/simetrica-bac
- [ ] `#786` **matsim** — ușoară
  https://www.pbinfo.ro/probleme/786/matsim
- [ ] `#213` **genmat4** — ușoară
  https://www.pbinfo.ro/probleme/213/genmat4