

Agenda

- Rezolvare tema
- Exerciții Matrici
- Tema

Rezolvare tema

1. Enunțul îl găsești în fișierul din sesiunea trecută.

```
#include<iostream>

using namespace std;
int main() {
    int n,x, contorX=0;
    cin >> n;
    int a[n];
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    cin >> x;
    for (int i =0; i < n; i++) {
        if (a[i] == x) {
            contorX++;
        }
    }

    cout << contorX << endl;

    return 0;
}
```

2. Enunțul îl găsești în fișierul din sesiunea trecută.

```
#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    int a[n];
    int frecventa[11] = {0};
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++) {
        frecventa[a[i]]++;
    }
}
```

```
        for (int i = 1; i < 11; i++) {  
            cout << i << " apare de " << frecventa[i] << " ori." << endl;  
        }  
  
        return 0;  
    }
```

3. Enuntul il gasesti in fisierul din sesiunea trecuta.

```
#include<iostream>  
  
using namespace std;  
int main() {  
    int n;  
    cin >> n;  
    int a[n];  
    int frecventa[21] = {0};  
    for (int i = 0; i < n; i++) {  
        cin >> a[i];  
    }  
    for (int i = 0; i < n; i++) {  
        frecventa[a[i]]++;  
    }  
    for (int i = 1; i < 21; i++) {  
        if (frecventa[i] == 1) {  
            cout << i << " ";  
        }  
    }  
  
    return 0;  
}
```

4. Enuntul il gasesti in fisierul din sesiunea trecuta.

```
#include<iostream>  
  
using namespace std;  
int main() {  
    int n;  
    cin >> n;  
    int a[n];  
    int frecventa[51] = {0};  
    int existaNumere = 0;  
    for (int i = 0; i < n; i++) {  
        cin >> a[i];  
    }  
    for (int i = 0; i < n; i++) {  
        frecventa[a[i]]++;  
    }  
}
```

```
    for (int i = 1; i < 51; i++) {
        if (frecventa[i] > n / 2) {
            cout << i << " ";
            existaNumere = 1;
        }
    }

    if (existaNumere == 0) {
        cout << " nu exista numere care sa apara mai mult de " << n/2 <<
" ori.";
    }

    return 0;
}
```

5. Enuntul il gasesti in fisierul din sesiunea trecuta.

```
#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    int frecventa[10] = {0};
    while (n) {
        int ultimaCifra = n % 10;
        frecventa[ultimaCifra]++;
        n /= 10;
    }

    for (int i = 0; i < 10; i++) {
        if (frecventa[i] > 0) {
            cout << "Cifra " << i << " apare de " << frecventa[i] << "
ori." << endl;
        }
    }

    return 0;
}
```

6. Enuntul il gasesti in fisierul de sesiunea trecuta

```
#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    int numere[n];
```

```
// Citim vectorul de la tastatura
for (int i = 0; i < n; i++) {
    cin >> numere[i];
}
// Initializam vectorul de frecventa cu 0 pentru toate numerele
int frecventa[11] = {0};

// Calculam frecventa pentru fiecare dintre elementele vectorului.
for (int i = 0; i < n; i++) {
    frecventa[numere[i]]++;
}

// Facem sortarea in functie de cerinta din enunt.
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n-1; j++) {
        // Daca numarul de pe pozitia j are o frecventa mai mica
        // decat numarul de pe pozitia urmatoare (j+1) atunci le
interschimbam
        if (frecventa[numere[j]] < frecventa[numere[j+1]]) {
            int temp = numere[j];
            numere[j] = numere[j+1];
            numere[j+1] = temp;
        }
        // Altfel daca numerele au aceeasi frecventa, si daca
        // numarul de pe pozitia curenta(j) este mai mare decat cel de pe pozitia
        // urmatoare (j+1) atunci le interschimbam
        else if (frecventa[numere[j]] == frecventa[numere[j+1]] &&
numere[j] > numere[j+1]) {
            int temp = numere[j];
            numere[j] = numere[j+1];
            numere[j+1] = temp;
        }
    }
}

// Afisam vectorul de frecventa ca sa ne verificam mai usor
for (int i = 0; i < 11; i++) {
    cout << i << " apare de " << frecventa[i] << " ori" << endl;
}

cout << "Vectorul de numere dupa sortarea in functie de frecventa:
";
for (int i = 0; i < n; i++) {
    cout << numere[i] << " ";
}

return 0;
}
```

7. Enuntul il gasesti in fisierul de sesiunea trecuta

```
#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    int numere[n];
    // Numerele sunt intre 1 si 20 deci avem nevoie ca vectorul de
frecventa
    // sa poata accesa frecventa[20] de aceea il declaram cu
frecventa[21];
    int frecventa[21]={0};

    // Citim cele n numere de la tastatura
    for (int i = 0; i < n; i++) {
        cin >> numere[i];
    }

    // Le calculam frecventa
    for (int i = 0; i < n; i++) {
        frecventa[numere[i]]++;
    }

    // Deoarece nu putem spune ca frecventa minima este egala cu
frecventa
    // primul numar deoarece acesta poate sa nu apara. Dam o valoare
care sigur nu poate fi adevarata
    // Adica n+1 si cum noi avem n numere in total, clar nu poate un
numar sa apara de n+1 ori.
    int frecventaMinima = n+1;

    // Acum cautam frecventa minima.
    for (int i = 0; i < 21; i++) {
        if (frecventa[i] > 0) {
            if (frecventa[i] < frecventaMinima) {
                frecventaMinima = frecventa[i];
            }
        }
    }

    // Pentru toate numerele care au frecventa minima, le afisam.
    // Atentie: i este numarul in sine si frecventa[i] este frecventa
numarului reprezentat de i
    for (int i = 0; i < 21; i++) {
        if (frecventa[i] == frecventaMinima) {
            cout << i << " ";
        }
    }

    return 0;
}
```

8. Enuntul il gasesti in fisierul de sesiunea trecuta

```
#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    int numere[n];
    // Numerele sunt intre 1 si 50 deci avem nevoie ca vectorul de
frecventa
    // sa poata accesa frecventa[50] de aceea il declaram cu
frecventa[51];
    int frecventa[51]={0};

    // Citim cele n numere de la tastatura
    for (int i = 0; i < n; i++) {
        cin >> numere[i];
    }

    // Le calculam frecventa
    for (int i = 0; i < n; i++) {
        frecventa[numere[i]]++;
    }

    // Deoarece nu putem spune ca frecventa minima este egala cu
frecventa
    // primul numar deoarece acesta poate sa nu apara. Dam o valoare
care sigur nu poate fi adevarata
    // Adica n+1 si cum noi avem n numere in total, clar nu poate un
numar sa apara de n+1 ori.
    int frecventaMinima = n+1;
    // Aici ne e usor sa initializam, deoarece setam frecventa maxima cu
0

    // si clar ca orice numar va avea frecventa mai mare decat 0.
    int frecventaMaxima = 0;

    // Ca sa nu mai parcurgem inca o data vectorul de frecventa sa
cautam
    // dupa numerele cu frecventa min, respectiv max, le initializam
    // de fiecare data cand gasim o noua frecventaMinima sau maxima.

    int numarFrecventaMinima;
    int numarFrecventaMaxima;
    // Acum cautam frecventa minima si maxima.
    for (int i = 0; i < 21; i++) {
        if (frecventa[i] > 0) {
            if (frecventa[i] < frecventaMinima) {
                frecventaMinima = frecventa[i];
                numarFrecventaMinima = i;
            }

            if (frecventa[i] > frecventaMaxima) {
```

```

        frecventaMaxima = frecventa[i];
        numarFrecventaMaxima = i;
    }
}

cout << numarFrecventaMinima << " apare de: " << frecventaMinima << endl;
cout << numarFrecventaMaxima << " apare de: " << frecventaMaxima << endl;

cout << "Diferenta dintre cele doua este de: ";
if (numarFrecventaMaxima > numarFrecventaMinima) {
    cout << numarFrecventaMaxima - numarFrecventaMinima << endl;
} else {
    cout << numarFrecventaMinima - numarFrecventaMaxima << endl;
}
return 0;
}

```

9. Enuntul il gasesti in fisierul de sesiunea trecuta

```

#include<iostream>

using namespace std;
int main() {
    int n;
    cin >> n;
    char text[n];
    cin>>text;
    char c;
    cin >> c;
    int contorC = 0;
    for (int i = 0; i < n; i++) {
        if (text[i] == c) {
            contorC++;
        }
    }
    cout << "Caracterul " << (char) c << " apare de " << contorC << " ori.";
    return 0;
}

```

10. Enuntul il gasesti in fisierul din sesiunea anterioara

```

#include<iostream>

using namespace std;
int main() {

```

```
int n;
cin >> n;
char text[n];
cin>>text;

int frecventa[26] = {0};

for (int i = 0; i < n; i++) {
    frecventa[text[i] - 'a']++;
}

int frecventaMaxima = 0;
for (int i = 0; i < 26; i++) {
    if (frecventa[i] > frecventaMaxima) {
        frecventaMaxima = frecventa[i];
    }
}

for (int i = 0; i < 26; i++) {
    if (frecventa[i] == frecventaMaxima) {
        cout << (char)(i+'a') << " ";
    }
}
return 0;
}
```

Exercitii de pe pbinfo:

1. <https://www.pbinfo.ro/probleme/666/nrprime>

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

int estePrim(int n);

int main() {
    int n, m;
    cin >> n >> m; // n => linii, m => coloane
    int matrice[n][m];
    int contor = 0;

    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            cin >> matrice[i][j];
        }
    }

    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++) {
            if ((i+1) % 2 == 0 && estePrim(matrice[i][j])) {
```



```
        contor++;
    }
}

cout << contor;

return 0;
}

int estePrim(int n) {
    if (n < 2) {
        return 0;
    }
    if (n == 2) {
        return 1;
    }
    if (n % 2 == 0) {
        return 0;
    }
    for (int i = 3; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}
```

2. <https://www.pbinfo.ro/probleme/4690/varfuri9>

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

int estePrim(int n);

int main() {
    int n, m;
    cin >> n >> m; // n => linii, m => coloane
    int matrice[n][m];
    int contor = 0;

    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            cin >> matrice[i][j];
        }
    }

    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++) {
            int sumaVecini = 0;
            // Daca suntem pe prima linie
            if (i == 0) {
```

```

        // Daca suntem pe prima coloana
        if (j == 0) {
            // inseamna ca avem vecini doar la Est si la Sud
            int vecinSud = matrice[i+1][j];
            int vecinEst = matrice[i][j+1];
            if (matrice[i][j] > (vecinEst + vecinSud)) {
                contor++;
            }
        }
        // Daca suntem pe ultima coloana
        else if (j == m-1) {
            // Inseamna ca avem vecini doar la sud si la vest
            int vecinSud = matrice[i+1][j];
            int vecinVest = matrice[i][j-1];
            if (matrice[i][j] > (vecinSud + vecinVest)) {
                contor++;
            }
        } else {
            // inseamna ca avem vecini in vest,est si sud
            int vecinSud = matrice[i+1][j];
            int vecinVest = matrice[i][j-1];
            int vecinEst = matrice[i][j+1];
            if (matrice[i][j] > (vecinSud + vecinVest+vecinEst))
        {
            contor++;
        }
    }
}
// Daca suntem pe ultima linie
else if (i == n-1) {
    // Daca suntem pe prima coloana
    if (j == 0) {
        // Inseamna ca avem vecini doar in nord si est
        int vecinNord = matrice[i-1][j];
        int vecinEst = matrice[i][j+1];
        if (matrice[i][j] > (vecinNord + vecinEst)) {
            contor++;
        }
    }
    // Daca suntem pe ultima coloana
    else if (j == m-1) {
        //Atunci inseamna ca avem vecini doar nord si vest
        int vecinNord = matrice[i-1][j];
        int vecinVest = matrice[i][j-1];
        if (matrice[i][j] > (vecinNord + vecinVest)) {
            contor++;
        }
    } else {
        //Inseamna ca avem vecini in nord si est si vest
        int vecinNord = matrice[i-1][j];
        int vecinVest = matrice[i][j-1];
        int vecinEst = matrice[i][j+1];
        if (matrice[i][j] > (vecinNord +
vecinVest+vecinEst)) {

```

```

        contor++;
    }
}
} else {
    // Inseamna ca nu suntem nici pe prima nici pe ultima
linie
    // Si daca suntem pe prima coloana
    if (j == 0) {
        // Atunci avem vecini in nord, sud si est
        int vecinNord = matrice[i-1][j];
        int vecinEst = matrice[i][j+1];
        int vecinSud = matrice[i+1][j];
        if (matrice[i][j] > (vecinSud + vecinEst +
vecinNord)) {
            contor++;
        }
    }
    // Daca suntem pe ultima coloana
    else if (j == m -1) {
        // Atunci inseamna ca avem vecini in nord,sud si
vest
        int vecinNord = matrice[i-1][j];
        int vecinVest = matrice[i][j-1];
        int vecinSud = matrice[i+1][j];
        if (matrice[i][j] > (vecinNord +
vecinSud+vecinVest)) {
            contor++;
        }
    } else {
        // Inseamna ca avem vecini in toate directiile
        int vecinNord = matrice[i-1][j];
        int vecinVest = matrice[i][j-1];
        int vecinSud = matrice[i+1][j];
        int vecinEst = matrice[i][j+1];
        if (matrice[i][j] > (vecinNord +
vecinVest+vecinSud+vecinEst)) {
            contor++;
        }
    }
}
}
}

cout << contor;

return 0;
}

int estePrim(int n) {
    if (n < 2) {
        return 0;
    }
    if (n == 2) {
        return 1;
    }
}

```

```
    }
    if (n % 2 == 0) {
        return 0;
    }
    for (int i = 3; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}
```

3. <https://www.pbinfo.ro/probleme/804/colgale>

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

int estePrim(int n);

int main() {
    int n, m;
    cin >> n >> m; // n => linii, m => coloane
    int matrice[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> matrice[i][j];
        }
    }

    int existaColoane = 0;
    for(int i = 0; i < m; i++) {
        int elementComun = matrice[0][i];
        int suntToateIdentice = 1;
        for(int j = 0; j < n; j++) {
            if (matrice[j][i] != elementComun) {
                suntToateIdentice = 0;
                break;
            }
        }
        if (suntToateIdentice == 1) {
            existaColoane = 1;
            cout << elementComun << " ";
        }
    }

    if (!existaColoane) {
        cout << "nu exista";
    }

    return 0;
}
```

Exercitii matrici

1. <https://www.pbinfo.ro/probleme/313/diagonale>

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {

    int n;
    cin >> n;
    int matrice[n][n];
    int sumaDiagonalaPrincipala = 0;
    int sumaDiagonalaSecundara = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrice[i][j];
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++){
            if (i == j) {
                sumaDiagonalaPrincipala+= matrice[i][j];
            }
            if ((i + j) == n-1) {
                sumaDiagonalaSecundara+= matrice[i][j];
            }
        }
    }

    cout << abs(sumaDiagonalaPrincipala - sumaDiagonalaSecundara);

    return 0;
}
```

2. <https://www.pbinfo.ro/probleme/780/cmmddsum>

```
#include <iostream>
#include <cmath>

int gcd(int a, int b);
```

```
using namespace std;

int main() {

    int n;
    cin >> n;
    int matrice[n][n];
    int sumaDeasupraDiagonalaPrincipala = 0;
    int sumaSubDiagonalaPrincipala = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrice[i][j];
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++){
            if (j > i) {
                sumaDeasupraDiagonalaPrincipala += matrice[i][j];
            }

            if (i > j) {
                sumaSubDiagonalaPrincipala += matrice[i][j];
            }
        }
    }

    cout << gcd(sumaSubDiagonalaPrincipala,
sumaDeasupraDiagonalaPrincipala);

    return 0;
}

int gcd(int a, int b) {
    while (b != 0) {
        int aux = b;
        b = a % b;
        a = aux;
    }
    return a;
}
```

3. <https://www.pbinfo.ro/probleme/786/matsim>

```
#include <iostream>

using namespace std;
```

```
int main() {  
  
    int n;  
    cin >> n;  
    int matrice[n][n];  
    int simetrica[n][n];  
  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            cin >> matrice[i][j];  
        }  
    }  
  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            simetrica[j][i] = matrice[i][j];  
        }  
    }  
  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            cout << simetrica[i][j] << " ";  
        }  
        cout << endl;  
    }  
  
    return 0;  
}
```

Tema

1. Se citește o matrice cu n linii și m coloane. Afișează elementele ei sub formă de tabel.

◦ Date de intrare:

```
n = 2, m = 3  
1 2 3  
4 5 6
```

◦ Date de iesire:

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

2. Se dă o matrice de dimensiune n x m. Calculează și afișează suma tuturor elementelor.

◦ Date de intrare:

```
n = 3, m = 4
1 2 3 4
4 5 6 7
8 9 10 11
```

- Date de iesire: 70

3. Se dă o matrice cu n linii și m coloane. Afișează suma elementelor de pe fiecare linie.

- Date de intrare:

```
n = 3, m = 2
1 2
3 4
5 6
```

- Date de iesire:

```
Linia 1: 3
Linia 2: 7
Linia 3: 11
```

4. Se dă o matrice de dimensiune n x m. Pentru fiecare coloană, afișează elementul maxim.

- Date de intrare:

```
n = 3, m = 3
1 5 2
7 3 8
4 6 9
```

- Date de iesire:

```
Coloana 1: 7
Coloana 2: 6
Coloana 3: 9
```

5. Se dă o matrice pătratică de ordin n. Afișează elementele de pe diagonala principală și suma lor.

- Date de intrare:


```
n = 3
1 2 3
4 5 6
7 8 9
```

- Date de iesire:

```
Elemente diagonala: 1 5 9
Suma = 15
```

6. Se dă o matrice pătratică de ordin n. Afișează elementele de pe diagonala secundară și produsul lor.

- Date de intrare:

```
n = 3
2 3 4
5 6 7
8 9 1
```

- Date de iesire:

```
Elemente diagonala secundară: 4 6 8
Produs = 192
```

7. Se dă o matrice pătratică de ordin n. Afișează separat elementele aflate strict deasupra și strict sub diagonala principală.

- Date de intrare:

```
n = 3
1 2 3
4 5 6
7 8 9
```

- Date de iesire:

```
Deasupra: 2 3 6
Sub: 4 7 8
```

8. Se dă o matrice pătratică de ordin n. Verifică dacă este simetrică față de diagonala principală (adică $M[i][j] = M[j][i]$).

◦ Date de intrare:

```
n = 3
1 2 3
2 4 5
3 5 6
```

◦ Date de iesire:

```
Matricea este simetrică
```

9. Se dă o matrice cu n linii și m coloane. Afișează transpusa ei (M^T), adică matricea obținută prin interschimbarea liniilor cu coloanele.

◦ Date de intrare:

```
n = 2, m = 3
1 2 3
4 5 6
```

◦ Date de iesire:

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

10. Se dau două matrici pătratice A și B , fiecare de ordin n . Calculează și afișează produsul lor $C = A \times B$.

◦ Date de intrare:


```
n = 2

1 2
3 4

5 6
7 8
```

◦ Date de iesire:

```
19 22
43 50
```

- 
11. <https://www.pbinfo.ro/probleme/1749/zona4>
 12. <https://www.pbinfo.ro/probleme/3124/patratmagic0>