Tablouri de date.

Conf. dr. Elena BAUTU
Facultatea de Matematică şi Informatică
Universitatea “Ovidius” Constanța

Tablouri (masive) de date

- Tablou (masiv de date, eng. *Data array*) = colecție *liniară* şi *omogenă* de date
  - Toate elementele au **acelaşi tip**,
  - dispuse **contiguu** în memorie, la **locații** (de memorie) **diferite**

- Declarație:

```
<tip elemente>  <nume vector> [lim1][lim2]…[limn]
```

  - Dimensiunile lim1, lim2,…limn sunt expresii constante
  - La intalnirea unei declaratii de tablou, compilatorul aloca o zona de memorie suficienta pentru a pastra valorile elementelor sale.

Tablouri (masive) de date

- Un tablou poate avea mai multe dimensiuni
  - o singură dimensiune → **vector** (sau şir)
    - **int v[5];** // un vector cu cinci numere întregi
  - 2 dimensiuni → **matrice** (tablou bi-dimensional)
    - float mat[2][3]; // o matrice cu 2x3 elemente
  - k dimensiuni (tablou de date de tipul *tip*)
    - tip nume[d1][d2]...[dk];
    - Exemplu: int cub[3][3][3];

Vector

Numele variabilei tablou este asociat cu adresa de inceput a tabloului.
Nume tablou = pointer

int v[10];

Am **declarat** o variabila vector v, cu componente de tip int

La **alocarea** unui vector, compilatorul nu efectuează **nici un fel de iniţializare** şi nu furnizează nici un mesaj de eroare dacă un element este folosit înainte de a fi iniţializat.

Un program corect va iniţializa, în orice caz, fiecare element înainte de a-l folosi.

v ⇔ &v[0]

Dimensiunea vectorului trebuie precizata la declarare, printr-o constanta numerica sau simbolica (e.g. definita cu #define). **#define N 10**

Accesul la elementele vectorului face uz de ***operatorul de indexare*** **[ ]:**

Elementul al 3-lea este cel de pe pozitia 2: v[2].

Intre [ ] se scrie *indicele* elementului, prin indice intelegand *deplasarea* fata de inceputul adresei de inceput (asociata variabilei v), masurata in numar de elemente de tipul precizat al vectorului.

Matrice

Declaram un tablou bidimensional, cu 4 linii si 3 coloane, cu elemente de tip int.

int mat[4][3];

mat[0][0]	mat[0][1]	mat[0][2]
mat[1][0]	mat[1][1]	mat[1][2]
mat[2][0]	mat[2][1]	mat[2][2]
mat[3][0]	mat[3][1]	mat[3][2]

De fapt, in memorie, tabloul este stocat liniar, astfel:

mat [0][0]	mat [0][1]	mat [0][2]	mat [1][0]	mat [1][1]	mat [1][2]	mat [2][0]	mat [2][1]	mat [2][2]	mat [3][0]	mat [3][1]	mat [3][2]

Accesul la un element al tabloului:
mat[1][2] – elementul de pe linia cu indice 1 si coloana cu indice 2

Important: numele unui tablou este pointer **constant**.
O consecinta imediata: numele unui vector NU se poate afla in partea stanga a operatorului de atribuire.

Copierea unui tablou peste alt tablou se va efectua prin copierea individuala, element cu element.

Tablouri - Stocarea informației

- numărul de elemente ale unui tablou cu k dimensiuni este egal cu **d1*d2*...*dk**

Definiți dimensiunile prin constante şi folosiți-le pe acestea în locul tastării explicite a valorilor în codul sursă.

```
#define N 3
#define M 4
```

- memorarea unui tablou cu tipul tip şi k dimensiuni necesită **sizeof(tip)*d1*d2*...*dk** bytes
  - Obs. Operatorul **sizeof(tip)** returnează nr de octeți necesari pentru memorarea unei variabile de tipul respectiv.

- In general, accesarea unui element al tabloului se face cu construcția :

  nume[i1][i2]…[in],

  unde i1, i2, …, in sunt coordonatele elementului (**0 ≤ ij < dimj, 1≤j≤k**).

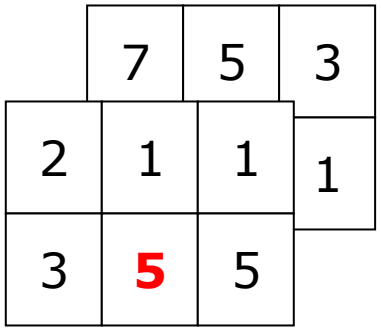
Tablou de date - elemente

10	5	6	8	9

A (**vector**)
-1 dimensiune
-5 elemente

7	5	3
4	6	**1**

B (**matrice**)
-2 dimensiuni
-6 elemente



C (**masiv** cu 3 dim.)
-3 dimensiuni
-12 elemente

A[3] = ?

B[1][2] = ?

C[0][1][1] = ?

```
int  A[5] ={10,5,6, 8, 9};
     cout<<A[3]; // 8
```

C[1][1][2]→1

Tablouri de date - elemente

- Numerotarea elementelor unui tablou incepe de la 0, pe fiecare dimensiune. Astfel, in tabloul :

  int Tab[5]={2,4,6,8,10};

- elementele sunt:

  Tab[0] -> 2;

  Tab[1] -> 4;

  Tab[2] -> 6;

  Tab[3] -> 8;

  Tab[4] ->10;

Inițializarea tablourilor

- Masivele pot fi **inițializate**:
  - **liniar**

tip nume[d_1][d_2]...[d_k]={v_1, v_2, ..., v_n};

valorile v_1, v_2, ..., v_n sunt stocate în masiv în ordine de la coordonatele mici (0,0,..., 0) la coordonatele mari

  - **pe axe**

tip nume[d_1][d_2]...[d_k]={{v_1,...,v_p}, {v_{p+1},...}, {...,v_n}};

fiecare grup de valori în acolade va fi stocată într-o dimensiune din masiv

Numele unui tablou reprezinta adresa de inceput a zonei de memorie care i-a fost alocata.

Inițializarea tablourilor

int A[10] = {1, 5, 2, 4}; // de la A[4] incolo, toate elementele sunt 0

1	5	2	4	0	0	0	0	0	0

- initializare liniara :

int A[2][5] = {1, 5, 2, 4};

1	5	2	4	0
0	0	0	0	0

- initializare pe axe:

1	0	0	0	0
3	0	0	0	0

int A[2][5] = {{1}, {3}};

Inițializarea tablourilor

- În cazul in care este **inițializat la declarare**, numărul de elemente din **prima** dimensiune a masivului **poate să lipsească**, iar el va fi stabilit de compilator.

  – int A[] = {1, 10, 5, 2}; //indicele ultimului element va fi 3

  este echivalent cu

  int A[4] = {1, 10, 5, 2};

  – int B[][3] = { {1,5}, {3}, { 2}};

  este echivalent cu

  int B[3][3] = {{1,5}, {3}, {2}};

1	5	0
3	0	0
2	0	0

  - int B[][3] = { 1, 5, 3, 2}; //sau int B[][3] = { 1,2, 3, 4, 5, 6, 7}; →3 linii

  este echivalent cu

  int B[2][3] = {{1,5, 3}, {2, 0, 0}};

Exemplu: citire/prelucrare/afisare vector

- Scrieți un program care citeşte un şir de cel mult 10 numere de la tastatură. Numarul de elemente este citit, de asemenea, de la tastatura.

- Programul va numara cate dintre elementele vectorului sunt numere pare, va calcula suma si produsul lor.

- Programul va afisa pe un rand elementele vectorului, separate printr-un spatiu si mesaje corespunzatoare pentru suma si produs.

Solutie propusa

```
//exemplu1.cpp
#define N 10
#include<iostream>
using namespace std;

int main(){
    int v[N];
    int n; // nr de elemente concrete
    cin >> n;
    int i;
    for(i=0; i<n; i++) // v[0]…v[n-1]
        cin >> v[i];

    int suma=0, produs=1;
    int contorPare=0;
    for(i=0;i<=n-1;i++){
        //i=poz curenta
        if(v[i] %2 == 0)
            contorPare++;
        suma += v[i];
        produs *= v[i];
    }
cout << "Vectorul este: ";
    for(i=0; i<n; i++)
        cout << v[i]<<" ";
    cout<< endl << "Sunt " <<
    contorPare << " elemente
    pare.\n";
    cout<< "Suma elementelor
    vectorului este "<<suma <<".\n";
    cout<< "Produsul elementelor
    vectorului este "<< produs <<".";

    return 0;
}
//V[0] * v[1] * v[2]…*v[n-1]
```

Generarea unui vector cu primele n (n<100) numere Fibonacci:

Citim n

Declaram vectorul cu dimensiune incapatoare

Initializam primii 2 termeni din sir cu primii 2 termeni ai sirului Fibonacci

Completam restul valorilor, folosind formula de recurenta cunoscuta

```
int main(){

}
```

Generarea unui vector cu primele n (n<100) numere Fibonacci:

Citim n

Declaram vectorul cu dimensiune incapatoare

Initializam primii 2 termeni din sir cu primii 2 termeni ai sirului Fibonacci

Completam restul valorilor, folosind formula de recurenta cunoscuta

```
#include<iostream>
using namespace std;
int main(){
    int n, i;
    cin>>n;

    long fib[100] = {1, 1};

    for(i=2; i<=n-1; i++){
        fib[i] = fib[i-1] + fib[i-2];
    }
    for(i=0; i<=n-1; i++)
        cout<<fib[i]<<" ";
} //exemplu4.cpp
```

Exemplu: citire/prelucrare/afisare matrice

Scrieți un program care citeşte de la intrarea standard un tablou bidimensional de cel mult 10x10 elemente numere intregi. Numarul de linii (n), este citit, de asemenea, de la tastatura. Numarul de coloane este egal cu numarul de linii.

Programul va numara cate dintre elementele matricei sunt numere *prime*, va calcula suma elementelor de pe diagonala principala si produsul elementelor de pe diagonala secundara.

A[3][3]; **A[linie][coloana]**
A[0][0] A[0][1] A[0][2]
A[1][0] **A[1][1]** A[1][2]
A[2][0] A[2][1] **A[2][2]**
Diagonala principala: linie = coloana
Diagonala secundara: linie + coloana = 2

Nr prim: un nr care se imparte doar cu el insusi si cu 1. Practic, vom cauta posibili divizori ai acestui nr., diferiti de 1 si de nr insusi.

Programul va afisa pe ecran elementele tabloului dispuse in forma unei matrice patratice.
Programul va afisa mesaje corespunzatoare pentru suma si produs.

Solutie propusa

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

int main(){
    int mat[10][10];
    int n;
    cin >>  n;;// n = dimensiunea matricei; 3
    int i, j, k;
    for(i=0; i<n; i++)// i - linia
        for(j=0; j<n; j++)//mat[0][0],
    mat[0][1], mat[0][2]
            cin >> mat[i][j];

    int suma=0, produs=1, contorPrime=0, prim
    = 1;
    for(i=0;i<n;i++)
        for(j=0; j<n; j++){//mat[i][j]
           //verificam nr prim
            prim = 1;
            for(k=2; k<=mat[i][j]/2; k++)
                if(mat[i][j] % k == 0 ){
                    prim = 0;
            }
            if(prim == 1){
                contorPrime++;
            }
        //verific diag. principala
              if(i==j)
                    suma += mat[i][j];
        // verific diag. secundara
                if(i+j==n-1)
                    produs *= mat[i][j];
        }
    cout << "Matricea este: \n";
        for(i=0; i<n; i++){
            for(j=0; j<n; j++)
                cout << mat[i][j]<<" ";
            cout<<endl;
        }

        cout<< endl << "Sunt " << contorPrime << "
        elemente prime.\n";
        cout<< "Suma elementelor de pe diagonala
        principala este "<<suma <<".\n";
        cout<< "Produsul elementelor de pe
        diagonala secundara este "<< produs <<".";

        return 0;
}

//exemplu2.cpp
```

Algoritmi fundamentali care lucreaza cu vectori

- *Maxim*, minim
- Numar de aparitii ale unui element
- Elemente distincte
- *Vector de aparitii (frecvente)*
- Sortare (bubble sort)
- Cautare secventiala/binara
- Inserare/stergere element in/din vector de pe prima/ultima/oarecare pozitie

Se citesc n numere intregi din intervalul [0, 100] de la tastatura.
Programul va afisa numarul care apare cel mai des in sirul de numere citite.

Idee:

Nu vom stoca numerele citite! Le prelucram pe masura ce le citim, unul cate unul.

Numerele citite sunt < 100. Vom folosi un vector de 100 de elemente, in care pe fiecare pozitie vom stoca numarul de aparitii al numarului corespunzatori pozitiei respective.

De exemplu:

- pe pozitia 9 → stocam nr de aparitii ale cifrei 9
- pe pozitia 56 → stocam nr de aparitii ale nr 56
- In general: pe pozitia **i**, stocam nr de aparitii ale nr **i** in sirul original

Se citesc n numere intregi din intervalul [0, 100] de la tastatura.
Programul va afisa numarul care apare cel mai des in sirul de numere citite.

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

int main(){
    int n, i;
    cin >> n;
    //initializam vect de frecvente cu 0
    int frecvente[100] = {0};
    /*in variabila element vom citi, rand pe rand, nr de la tastatura*/
    int element;
    for(i=1; i<=n; i++){
        cin>>element;
          frecvente[element]++;
    }
int maxim = 0, pozitie = 0;
    for (i = 10;i < 100; i++)
            if(frecvente[i] > maxim){
                    maxim = frecvente[i];
                    pozitie = i;
            }

    cout<<"Elementul "<<pozitie<<" apare de "<<maxim<<" ori.";

    return 0;
}

//exemplu3.cpp
```

Exercitii cu vectori

2. Tablourile unidimensionale **A** și **B** au elementele: **A=(2,20,27,36,50)**, iar **B=(63,45,8,5,3)**. În urma interclasării lor în ordine descrescătoare se obține tabloul cu elementele:

a. (63,45,27,36,50)

b. (63,45,8,5,3,2,20,27,36,50)

c. (63,2,45,20,27,8,36,5,50,3)

d. (63,50,45,36,27,20,8,5,3,2)

2. Tablourile unidimensionale **A** și **B** au elementele: **A=(2,20,27,36,50)**, iar **B=(63,45,8,5,3)**. În urma interclasării lor în ordine descrescătoare se obține tabloul cu elementele:

a. (63,45,27,36,50)
b. (63,45,8,5,3,2,20,27,36,50)
c. (63,2,45,20,27,8,36,5,50,3)
d. (63,50,45,36,27,20,8,5,3,2)

Interclasare:

Doi vectori de numere, sortate in ordine crescatoare, se unesc intr-un al treilea vector, a.i. elementele in vectorul final sa fie in ordine crescatoare.

La fiecare pas, se memoreaza in al treilea vector, un element din unul din ceilalti 2 vectori, care se potriveste a fi adaugat, asa incat sa pastreze ordinea.

Raspuns: 63, 50, 45, 36, 27, 20, 8, 5, 3, 2

3. Pentru a verifica dacă în tabloul unidimensional **(4,8,9,14,16,24,48)** există elementul cu valoarea **x** se aplică metoda căutării binare. Știind că valoarea **x** a fost comparată cu două elemente ale tabloului pe parcursul aplicării metodei, două valori ale lui **x** ar putea fi:

a. 4, 8 b. 8, 24 c. 9, 16 d. 24, 48

Cautare: intr-un vector de N elemente, se cauta un nr n.

Cautare secventiala:
Se parcurge vectorul, element cu element, comparandu-se de fiecare data elementul de la pozitia curenta cu nr cautat.

Cautare binara:
Vectorul trebuie sa fie **sortat** in ordine crescatoare.

Cautarea se efectueaza intre 2 pozitii, memorate in 2 variabile: s (initial s=0), d (initial d=N-1).

Nr cautat (n) se compara cu elementul din mijlocul vectorului (indicele lui este *m=s+(d-s)/2*) .
- dc este egal → se termina cautarea
- dc n < elementul de la pozitia poz → d = poz - 1 → cautarea va continua in portiunea de la stanga pozitiei poz)
- dc n > elementul de la pozitia poz → s = poz + 1 → cautarea va continua in portiunea de la dreapta pozitiei poz

Raspuns:
- cautarea lui 4 → comparatie cu 14, comparatie cu 8, comparatie cu 4 : 3 comparatii
- cautarea lui 8 → comparatie cu 14, comparatie cu 8 : 2 comparatii
- cautarea lui 24 → comparatie cu 14, comparatie cu 24 : 2 comparatii

2. Tablourile unidimensionale **A** și **B** au elementele: **A=(2,20,27,36,50)**, iar **B=(63,45,8,5,3)**. În urma interclasării lor în ordine crescătoare se obține tabloul cu elementele:

a. **(2,3,5,8,20,27,36,45,50,63)**

b. **(2,20,8,5,3)**

c. **(2,20,27,36,50,3,5,8,45,63)**

d. **(2,63,20,45,8,27,5,36,3,50)**

3. Pentru a verifica dacă într-un tablou unidimensional există elementul cu valoarea **x=3**, se aplică metoda căutării binare, iar succesiunea de elemente a căror valoare se compară cu **x** pe parcursul aplicării metodei este **14**, **8**, **4**. Elementele tabloului pot fi:

a. **(4,8,9,14,16,24,48)**

b. **(14,14,8,8,4,4)**

c. **(14,8,4,3,2,0)**

d. **(48,14,9,8,7,4,2)**

2. Tablourile unidimensionale **A** și **B** au elementele: **A=(2,20,27,36,50)**, iar **B=(63,45,8,5,3)**. În urma interclasării lor în ordine crescătoare se obține tabloul cu elementele:

a. **(2,3,5,8,20,27,36,45,50,63)**
b. **(2,20,8,5,3)**
c. **(2,20,27,36,50,3,5,8,45,63)**
d. **(2,63,20,45,8,27,5,36,3,50)**

3. Pentru a verifica dacă într-un tablou unidimensional există elementul cu valoarea **x=3**, se aplică metoda căutării binare, iar succesiunea de elemente a căror valoare se compară cu **x** pe parcursul aplicării metodei este **14**, **8**, **4**. Elementele tabloului pot fi:

a. **(4,8,9,14,16,24,48)**
b. **(14,14,8,8,4,4)**
c. **(14,8,4,3,2,0)**
d. **(48,14,9,8,7,4,2)**

Raspuns:

2. 2, 3, 5, 8, 20, 27, 36, 45, 50, 63 → a.

2. Tablourile unidimensionale **A** și **B** au elementele: **A=(2,20,27,36,50)**, iar **B=(63,45,8,5,3)**. În urma interclasării lor în ordine crescătoare se obține tabloul cu elementele:

a. (2,3,5,8,20,27,36,45,50,63)
b. (2,20,8,5,3)
c. (2,20,27,36,50,3,5,8,45,63)
d. (2,63,20,45,8,27,5,36,3,50)

3. Pentru a verifica dacă într-un tablou unidimensional există elementul cu valoarea **x=3**, se aplică metoda căutării binare, iar succesiunea de elemente a căror valoare se compară cu **x** pe parcursul aplicării metodei este **14**, **8**, **4**. Elementele tabloului pot fi:

a. (4,8,9,14,16,24,48)
b. (14,14,8,8,4,4)
c. (14,8,4,3,2,0)
d. (48,14,9,8,7,4,2)

Raspuns:

2. 2, 3, 5, 8, 20, 27, 36, 45, 50, 63 → a.

3. a. 7 elemente in tablou;

comparatii cu: 14, 8, 4

Obs. b.,c. → sortati descrescatori

d. → valori neordonate

3. În urma interclasării în ordine descrescătoare a tablourilor unidimensionale **A** şi **B** se obține tabloul: **(38,38,30,25,25,13,12,10,8,7,5)**. Scrieți un exemplu de valori memorate în tablourile **A** şi **B**, în ordinea apariției lor în tablou. **(6p.)**

3. În urma interclasării în ordine descrescătoare a tablourilor unidimensionale **A** şi **B** se obține tabloul: **(38,38,30,25,25,13,12,10,8,7,5)**. Scrieți un exemplu de valori memorate în tablourile **A** şi **B**, în ordinea apariției lor în tablou. **(6p.)**

(38, 38, 30, 25,25,13,12,10,8,7,5)

A: (38, 30, 25,13,10,7,5)

B: (38, 25, 12, 8)

2. Pentru a verifica dacă în tabloul unidimensional **(2,8,13,19,20,38,47)** există elementul cu valoarea **x** se aplică metoda căutării binare. Știind că valoarea **x** este comparată cu trei elemente ale tabloului pe parcursul aplicării metodei, indicați o valoare cu care **x** NU poate fi egală.

a. 2 **b.** 8 **c.** 20 **d.** 47

3. Variabilele **i** și **j** sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran valorile alăturate, în această ordine.

```
for(i=1;i<=4;i++)
{ for(j=1;j<=5;j++)
    cout<<.........<<" "; | printf("%d ",.........);
  cout<<endl;  |  printf("\n");
}
```

```
1 2 3 4 5
2 4 6 8 0
3 6 9 2 5
4 8 2 6 0
```

a. (i+j)%10 **b.** (i*j)%10 **c.** i+j%5 **d.** i*j%5

2. Pentru a verifica dacă în tabloul unidimensional **(2,8,13,19,20,38,47)** există elementul cu valoarea **x** se aplică metoda căutării binare. Știind că valoarea **x** este comparată cu trei elemente ale tabloului pe parcursul aplicării metodei, indicați o valoare cu care **x** NU poate fi egală.

a. 2 **b.** 8 **c.** 20 **d.** 47

3. Variabilele **i** și **j** sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran valorile alăturate, în această ordine.

```
for(i=1;i<=4;i++)
{ for(j=1;j<=5;j++)
    cout<<.........<<" "; | printf("%d ",.........);
  cout<<endl;  |  printf("\n");
}
```

```
1 2 3 4 5
2 4 6 8 0
3 6 9 2 5
4 8 2 6 0
```

a. (i+j)%10 **b.** (i*j)%10 **c.** i+j%5 **d.** i*j%5

2. (2, 8, 13, 19, 20, 38, 47)
Prin cautare binara, se vor efectua comparatii astfel:
-intai, cu valoarea din mijloc: 19
-Apoi: fie la stanga, cu 8, fie la dreapta, cu 38
-Apoi: fie cu 2 sau 13, fie cu 20 sau 47
Pentru 2, 20, 47 – se efectueaza 3 comparatii
Daca numarul cautat ar fi 8, rezultatul ar fi obtinut din doar 2 comparatii.

2. Pentru a verifica dacă în tabloul unidimensional **(2,8,13,19,20,38,47)** există elementul cu valoarea **x** se aplică metoda căutării binare. Știind că valoarea **x** este comparată cu trei elemente ale tabloului pe parcursul aplicării metodei, indicați o valoare cu care **x** NU poate fi egală.

a. 2 **b.** 8 **c.** 20 **d.** 47

3. Variabilele **i** și **j** sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze pe ecran valorile alăturate, în această ordine.

```
for(i=1;i<=4;i++)
{ for(j=1;j<=5;j++)
    cout<<.........<<" "; | printf("%d ",.........);
  cout<<endl;  |  printf("\n");
}
```

```
1 2 3 4 5
2 4 6 8 0
3 6 9 2 5
4 8 2 6 0
```

a. (i+j)%10 **b.** (i*j)%10 **c.** i+j%5 **d.** i*j%5

2.
Prin cautare binara, se vor efectua comparatii astfel:
-intai, cu valoarea din mijloc: 19
-Apoi: fie la stanga, cu 8, fie la dreapta, cu 38
-Apoi: fie cu 2 sau 13, fie cu 20 sau 47
Pentru 2, 20, 47 – se efectueaza 3 comparatii
Daca numarul cautat ar fi 8, rezultatul ar fi obtinut din doar 2 comparatii.

3. Aleg o pozitie, din interiorul matricei: de exemplu, pozitia incercuita. Valoarea 9 este afisata pentru i=3 si j = 3. Verific formulele: (3+3)%10 →6 ; **(3*3)%10→9;** 3+3%5→6; 3*3%5→4
Observatie. Operatorii * si % au aceeasi prioritate

3. Tabloul unidimensional **X** are elementele **(5,9,10,13,19)**, iar în urma interclasării lui în ordine crescătoare cu tabloul **Y** se obține tabloul **(5,8,9,10,12,13,19,19,30,52)**. Indicați elementele tabloului **Y**, în ordinea apariției lor în acesta.

a. **(52,30,19,12,8)**

b. **(36,20,12,6,6)**

c. **(5,9,10,13,19,8,12,19,30,52)**

d. **(5,6,9,6,10,12,13,20,19,36)**

4. Pentru a verifica dacă în tabloul unidimensional **(2,5,8,11,16,22,40)** există elementul cu valoarea **x=16** se aplică metoda căutării binare. Indicați succesiunea de elemente a căror valoare se compară cu **x** pe parcursul aplicării metodei.

a. **11,16**

b. **16**

c. **11,22,16**

d. **22,16**

3. Tabloul unidimensional **X** are elementele **(5,9,10,13,19)**, iar în urma interclasării lui în ordine crescătoare cu tabloul **Y** se obține tabloul **(5,8,9,10,12,13,19,19,30,52)**. Indicați elementele tabloului **Y**, în ordinea apariției lor în acesta.

a. (52,30,19,12,8)
b. (36,20,12,6,6)
c. (5,9,10,13,19,8,12,19,30,52)
d. (5,6,9,6,10,12,13,20,19,36)

4. Pentru a verifica dacă în tabloul unidimensional **(2,5,8,11,16,22,40)** există elementul cu valoarea **x=16** se aplică metoda căutării binare. Indicați succesiunea de elemente a căror valoare se compară cu **x** pe parcursul aplicării metodei.

a. 11,16
b. 16
c. 11,22,16
d. 22,16

3.

(5, 8, 9, 10, 12, 13, 19, 19, 30, 52)→ Y (8, 12, 19, 30, 52) → descrescator

Raspuns a.

3. Tabloul unidimensional `X` are elementele `(5,9,10,13,19)`, iar în urma interclasării lui în ordine crescătoare cu tabloul `Y` se obține tabloul `(5,8,9,10,12,13,19,19,30,52)`. Indicați elementele tabloului `Y`, în ordinea apariției lor în acesta.

a. `(52,30,19,12,8)`
b. `(36,20,12,6,6)`
c. `(5,9,10,13,19,8,12,19,30,52)`
d. `(5,6,9,6,10,12,13,20,19,36)`

4. Pentru a verifica dacă în tabloul unidimensional `(2,5,8,11,16,22,40)` există elementul cu valoarea `x=16` se aplică metoda căutării binare. Indicați succesiunea de elemente a căror valoare se compară cu `x` pe parcursul aplicării metodei.

a. `11,16`
b. `16`
c. `11,22,16`
d. `22,16`

3.
(5, 8, 9, 10, 12, 13, 19, 19, 30, 52)→ Y (8, 12, 19, 30, 52) → descrescator
Raspuns a.

4.
Prima comparatie: cu 11
A doua comparatie: cu 22
A treia comparatie: cu 16
Raspuns: c

2. O valoare **filtrează** un șir dacă există doi termeni ai șirului care au acea valoare, unul fiind în prima jumătate a șirului, iar celălalt în a doua jumătate a șirului.

Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul `[2,20]`: `n` și un șir de `2·n` numere, elemente ale unui tablou unidimensional, cu proprietatea că atât primele `n`, cât și ultimele `n` sunt distincte. Programul afișează pe ecran valorile care pot filtra șirul, într-o ordine oarecare, separate prin câte un spațiu, sau mesajul `nu exista`, dacă nu există astfel de valori.

Exemplu: pentru `n=4` și tabloul (`4,5,7,2,2,6,4,7`) se afișează pe ecran, nu neapărat în această ordine, numerele `2 4 7` **(10p.)**

2. O valoare **filtrează** un șir dacă există doi termeni ai șirului care au acea valoare, unul fiind în prima jumătate a șirului, iar celălalt în a doua jumătate a șirului.

Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul `[2,20]`: `n` și un șir de `2·n` numere, elemente ale unui tablou unidimensional, cu proprietatea că atât primele `n`, cât și ultimele `n` sunt distincte. Programul afișează pe ecran valorile care pot filtra șirul, într-o ordine oarecare, separate prin câte un spațiu, sau mesajul `nu exista`, dacă nu există astfel de valori.

Exemplu: pentru `n=4` și tabloul (`4,5,7,2,2,6,4,7`) se afișează pe ecran, nu neapărat în această ordine, numerele `2 4 7` **(10p.)**

Idee:

- Citim datele
- Initializam un contor al valorilor care filtreaza: contor = 0;
- Observam ca valorile comune sunt afisate in ordinea in care se gasesc in a 2-a jumatate
- Parcurgem ultimele n pozitii din vector. Pentru fiecare element, parcurgem primele n pozitii din vector
  - Pentru fiecare pereche, verificam daca elementul din ultima jumatate coincide cu elem din prima jumatate.
  - Daca da – il afisam, incrementam contorul
- La final, daca contor a ramas 0, afisam mesaj “nu exista”

2. O valoare **filtrează** un șir dacă există doi termeni ai șirului care au acea valoare, unul fiind în prima jumătate a șirului, iar celălalt în a doua jumătate a șirului.

Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul **[2,20]**: **n** și un șir de **2·n** numere, elemente ale unui tablou unidimensional, cu proprietatea că atât primele **n**, cât și ultimele **n** sunt distincte. Programul afișează pe ecran valorile care pot filtra șirul, într-o ordine oarecare, separate prin câte un spațiu, sau mesajul **nu exista**, dacă nu există astfel de valori.

Exemplu: pentru **n=4** și tabloul (**4,5,7,2,2,6,4,7**) se afișează pe ecran, nu neapărat în această ordine, numerele **2 4 7** **(10p.)**

Idee:
- Citim datele
- Initializam un contor al valorilor care filtreaza: contor = 0;
- Observam ca valorile comune sunt afisate in ordinea in care se gasesc in a 2-a jumatate
- Parcurgem ultimele n pozitii din vector. Pentru fiecare element, parcurgem primele n pozitii din vector
  - Pentru fiecare pereche, verificam daca elementul din ultima jumatate coincide cu elem din prima jumatate.
  - Daca da – il afisam, incrementam contorul
- La final, daca contor a ramas 0, afisam mesaj “nu exista”

```
#include<iostream>
using namespace std;
int main(){
    int n, v[40];
    cin>>n;
    int i, j;
    for(i=0; i<=2*n-1; i++)
        cin>>v[i];

    int contor = 0;
    for(i = n; i<=2*n-1; i++)
        for(j=0; j<=n-1; j++)
            if(v[i] == v[j]){
                contor ++;
                cout<<v[i]<<endl;
            }
    if(contor == 0)
        cout<<"nu exista"<<endl;
} // 14.cpp
```

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** ($n \in [2, 10^2]$) și construiește în memorie un tablou unidimensional cu **n** elemente, cu proprietatea că parcurgându-l de la stânga la dreapta se obține șirul primelor **n** pătrate perfecte pare, ordonat strict descrescător, ca în exemplu. Elementele tabloului obținut se afișează pe ecran, separate prin câte un spațiu.
Exemplu: pentru **n=6** se obține tabloul **(100,64,36,16,4,0)**. **(10p.)**

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** ($n \in [2, 10^2]$) și construiește în memorie un tablou unidimensional cu **n** elemente, cu proprietatea că parcurgându-l de la stânga la dreapta se obține șirul primelor **n** pătrate perfecte pare, ordonat strict descrescător, ca în exemplu. Elementele tabloului obținut se afișează pe ecran, separate prin câte un spațiu.
Exemplu: pentru **n=6** se obține tabloul **(100,64,36,16,4,0).** **(10p.)**

Idee:
-declaram un vector de 100 de elemente intregi (100 este o dimensiune acoperitoare pentru cerinta problemei).

-Parcurgem vectorul pozitie cu pozitie, de la ultima pozitie catre prima si completam cu elemente generate conform cerintei

-Memoram si un nr care va reprezenta numarul par curent, al carui patrat il stocam in vector.

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** ($n \in [2, 10^2]$) și construiește în memorie un tablou unidimensional cu **n** elemente, cu proprietatea că parcurgându-l de la stânga la dreapta se obține șirul primelor **n** pătrate perfecte pare, ordonat strict descrescător, ca în exemplu. Elementele tabloului obținut se afișează pe ecran, separate prin câte un spațiu.
Exemplu: pentru **n=6** se obține tabloul **(100,64,36,16,4,0)**. **(10p.)**

Idee:
-declaram un vector de 100 de elemente intregi (100 este o dimensiune acoperitoare pentru cerinta problemei).

-Parcurgem vectorul pozitie cu pozitie, de la ultima pozitie catre prima si completam cu elemente generate conform cerintei
-Memoram si un nr care va reprezenta numarul par curent, al carui patrat il stocam in vector.

```
//1.cpp
#include<iostream>
using namespace std;
int main(){
    int n; cin>>n;
    int v[100];
    int i, nrpar = 0, x=0;
    for(i=n-1; i>=0; i--){
        v[i] = nrpar*nrpar;
        nrpar += 2;
    }
    for(i=0; i<=n-1; i++)
        cout<<v[i]<<" ";
}
```

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** (**n∈[2,10²]**) și cele **2·n** elemente ale unui tablou unidimensional, numere naturale din intervalul **[1,10⁹]**. Programul afișează pe ecran, separate prin câte un spațiu, primele **n** elemente ale tabloului, parcurse de la stânga la dreapta, urmate de ultimele **n** elemente ale tabloului, parcurse de la dreapta la stânga.

Exemplu: pentru **n=5** și tabloul **(1,2,3,4,5,3,1,8,6,4)** se afișează pe ecran numerele

1 2 3 4 5 4 6 8 1 3 **(10p.)**

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** ($n \in [2, 10^2]$) și cele **2·n** elemente ale unui tablou unidimensional, numere naturale din intervalul $[1, 10^9]$. Programul afișează pe ecran, separate prin câte un spațiu, primele **n** elemente ale tabloului, parcurse de la stânga la dreapta, urmate de ultimele **n** elemente ale tabloului, parcurse de la dreapta la stânga.
Exemplu: pentru **n=5** și tabloul **(1,2,3,4,5,3,1,8,6,4)** se afișează pe ecran numerele
1 2 3 4 5 4 6 8 1 3 **(10p.)**

- Declaram un vector cu 200 de elemente
- Parcurgem intai primele n elemente utile, de la stanga la dreapta (in ordine crescatoare a indicelui in vector), ale vectorului si le afisam
- Parcurgem apoi ultimele n elemente utile, in ordine descrescatoare a indicelui lor in vector, si le afisam

2. Scrieți un program C/C++ care citește de la tastatură un număr natural **n** (**n∈[2,10²]**) și cele **2·n** elemente ale unui tablou unidimensional, numere naturale din intervalul **[1,10⁹]**. Programul afișează pe ecran, separate prin câte un spațiu, primele **n** elemente ale tabloului, parcurse de la stânga la dreapta, urmate de ultimele **n** elemente ale tabloului, parcurse de la dreapta la stânga.
Exemplu: pentru **n=5** și tabloul **(1,2,3,4,5,3,1,8,6,4)** se afișează pe ecran numerele
1 2 3 4 5 4 6 8 1 3 **(10p.)**

- Declaram un vector cu 200 de elemente
- Parcurgem intai primele n elemente utile, de la stanga la dreapta (in ordine crescatoare a indicelui in vector), ale vectorului si le afisam
- Parcurgem apoi ultimele n elemente utile, in ordine descrescatoare a indicelui lor in vector, si le afisam

```
//3.cpp
#include<iostream>
using namespace std;
int main(){
    int v[200];
    int n, i;
    cin>>n;

    for(i=0; i<=2*n-1; i++)
        cin>>v[i];

    for(i=0;i<=n-1;i++)
        cout<<v[i]<<" ";

    for(i=2*n-1; i>=n; i--)
        cout<<v[i]<<" ";

        return 0;
}
```

Exercitii cu tablouri bidimensionale (matrice)

2. Scrieți un program C/C++ care citește de la tastatură două numere naturale din intervalul $[2,10^2]$, `m` și `n`, și construiește în memorie un tablou bidimensional cu `m` linii și `n` coloane, cu proprietatea că parcurgându-l linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, se obține șirul primelor `m*n` **pătrate perfecte pare**, ordonat strict descrescător, ca în exemplu.

Elementele tabloului obținut se afișează pe ecran, fiecare linie a tabloului pe câte o linie a ecranului, valorile de pe aceeași linie fiind separate prin câte un spațiu.
Exemplu: pentru `m=2`, `n=3` se obține tabloul alăturat. **(10p.)**

```
100  64  36
 16   4   0
```

2. Scrieți un program C/C++ care citește de la tastatură două numere naturale din intervalul $[2, 10^2]$, **m** și **n**, și construiește în memorie un tablou bidimensional cu **m** linii și **n** coloane, cu proprietatea că parcurgându-l linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, se obține șirul primelor **m*n pătrate perfecte pare**, ordonat strict descrescător, ca în exemplu.

Elementele tabloului obținut se afișează pe ecran, fiecare linie a tabloului pe câte o linie a ecranului, valorile de pe aceeași linie fiind separate prin câte un spațiu.
Exemplu: pentru **m=2**, **n=3** se obține tabloul alăturat. **(10p.)**

```
100  64  36
 16   4   0
```

Idee:

Parcurgem matricea de jos in sus, din coltul din dreapta catre coltul din stanga

In coltul dreapta jos stocam patratul lui 0

Ulterior, obtinem urmatoarele numere pare si le stocam in continuare

2. Scrieți un program C/C++ care citește de la tastatură două numere naturale din intervalul $[2,10^2]$, **m** și **n**, și construiește în memorie un tablou bidimensional cu **m** linii și **n** coloane, cu proprietatea că parcurgându-l linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, se obține șirul primelor **m*n pătrate perfecte pare**, ordonat strict descrescător, ca în exemplu.

Elementele tabloului obținut se afișează pe ecran, fiecare linie a tabloului pe câte o linie a ecranului, valorile de pe aceeași linie fiind separate prin câte un spațiu.
Exemplu: pentru **m=2**, **n=3** se obține tabloul alăturat. **(10p.)**

```
100  64  36
 16   4   0
```

Idee:

Parcurgem matricea de jos in sus, din coltul din dreapta catre coltul din stanga

In coltul dreapta jos stocam patratul lui 0

Ulterior, obtinem urmatoarele numere pare si le stocam in continuare

```
//2.cpp
#include<iostream>
using namespace std;
int main(){
    int mat[100][100];
     int m, n, i, j, nr=0;;
    cin>>m>>n;
    for(i=m-1; i>=0; i--)
        for(j=n-1; j>=0; j--)
        {
            mat[i][j] = nr*nr;
            nr+=2;
        }


    for(i=0; i<=m-1; i++){
        for(j=0; j<=n-1; j++){
            cout<<mat[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

2. Scrieți un program C/C++ care citește de la tastatură numărul natural **n** ($n \in [2, 10^2]$) și elementele unui tablou bidimensional cu **n** linii și **n** coloane, numere naturale din intervalul $[0, 10^9]$.

Programul afișează pe ecran, separate prin câte un spațiu, elementele primului pătrat concentric, parcurs în sens invers al acelor de ceasornic, începând din colțul său stânga-sus, ca în exemplu. Primul pătrat concentric este format din prima și ultima linie, prima și ultima coloană a tabloului.

Exemplu: pentru **n=5** și tabloul alăturat, se afișează pe ecran numerele

1 2 3 4 5 6 7 8 9 0 2 4 6 8 1 3 **(10p.)**

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

2. Scrieți un program C/C++ care citește de la tastatură numărul natural **n** ($n \in [2, 10^2]$) și elementele unui tablou bidimensional cu **n** linii și **n** coloane, numere naturale din intervalul $[0, 10^9]$.

Programul afișează pe ecran, separate prin câte un spațiu, elementele primului pătrat concentric, parcurs în sens invers al acelor de ceasornic, începând din colțul său stânga-sus, ca în exemplu. Primul pătrat concentric este format din prima și ultima linie, prima și ultima coloană a tabloului.

Exemplu: pentru **n=5** și tabloul alăturat, se afișează pe ecran numerele

1 2 3 4 5 6 7 8 9 0 2 4 6 8 1 3 **(10p.)**

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

- Declaram tablou bidimensional si citim elementele sale de la tastatura
- Parcurgem tabloul, element cu element, de la prima la ultima linie, de la stanga la dreapta pe fiecare linie
- La fiecare pozitie, verificam daca pozitia se afla pe conturul matricei (prima/ultima linie, prima/ultima coloana)

2. Scrieți un program C/C++ care citește de la tastatură numărul natural **n** ($n \in [2, 10^2]$) și elementele unui tablou bidimensional cu **n** linii și **n** coloane, numere naturale din intervalul $[0, 10^9]$.

Programul afișează pe ecran, separate prin câte un spațiu, elementele primului pătrat concentric, parcurs în sens invers al acelor de ceasornic, începând din colțul său stânga-sus, ca în exemplu. Primul pătrat concentric este format din prima și ultima linie, prima și ultima coloană a tabloului.

Exemplu: pentru **n=5** și tabloul alăturat, se afișează pe ecran numerele

1 2 3 4 5 6 7 8 9 0 2 4 6 8 1 3 **(10p.)**

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

- Declaram tablou bidimensional si citim elementele sale de la tastatura
- Parcurgem tabloul, element cu element, de la prima la ultima linie, de la stanga la dreapta pe fiecare linie
- La fiecare pozitie, verificam daca pozitia se afla pe conturul matricei (prima/ultima linie, prima/ultima coloana)

```
#include<iostream>
using namespace std;
int main(){
    int mat[100][100];
    int n, i, j;
    cin>>n;

    for(i=0; i<=n-1; i++)
        for(j=0; j<=n-1; j++)
            cin>>mat[i][j];
    for(i=0;i<=n-1;i++){
        for(j=0; j<=n-1; j++)
            if(i==0 || i==n-1 || j==0 || j==n-1)
                cout<<mat[i][j];
            else cout<<" ";
        cout<<endl;
        }

    return 0;
} //4.cpp
```

2. Scrieţi un program C/C++ care citeşte de la tastatură două numere naturale din intervalul `[2,20]`, `n` și `k`, şi construieşte în memorie un tablou bidimensional cu `n` linii şi `n·k` coloane, numerotate începând cu `1`, astfel încât fiecare linie `i` (`i∈[1,n]`) memorează un şir crescător de termeni cu proprietatea că primul termen este `i`, fiecare valoare apare în şir de exact `k` ori și oricare doi termeni alăturați au valori egale sau consecutive.

Programul afişează pe ecran tabloul construit, fiecare linie a tabloului pe câte o linie a ecranului, cu valorile aflate pe aceeaşi linie separate prin câte un spaţiu.
Exemplu: dacă `n=4` și `k=3`, se afişează pe ecran tabloul alăturat.
(10p.)

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

Ideea: umplem n grupuri de cate k elemente pe fiecare linie.
Observam ca fiecare linie i incepe cu i+1+0, si apoi la fiecare k elemente, se adauga si j (care spune la al catelea grup s-a ajuns pe linia respectiva).

2. Scrieţi un program C/C++ care citeşte de la tastatură două numere naturale din intervalul `[2,20]`, `n` și `k`, şi construieşte în memorie un tablou bidimensional cu `n` linii şi `n·k` coloane, numerotate începând cu `1`, astfel încât fiecare linie `i` (`i∈[1,n]`) memorează un şir crescător de termeni cu proprietatea că primul termen este `i`, fiecare valoare apare în şir de exact `k` ori și oricare doi termeni alăturați au valori egale sau consecutive.

Programul afişează pe ecran tabloul construit, fiecare linie a tabloului pe câte o linie a ecranului, cu valorile aflate pe aceeaşi linie separate prin câte un spaţiu.
Exemplu: dacă `n=4` și `k=3`, se afişează pe ecran tabloul alăturat.
(10p.)

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

Ideea: umplem n grupuri de cate k elemente pe fiecare linie.
Observam ca fiecare linie i incepe cu i+1+0, si apoi la fiecare k elemente, se adauga si j (care spune la al catelea grup s-a ajuns pe linia respectiva).

```
#include<iostream>
using namespace std;
int main(){
  int n, k;
  cin>>n>>k;
  int a[100][100];
  int i=0, j, l;

  for(i=0; i<n; i++)
      for(j=0; j<n; j++)
          for(l=0; l<=k-1; l++)
            a[i][j*k+l]=i+1 +j;
}
```

```
for(i=0;i<n;i++){
        for(j=0;j<n*k; j++)
            cout<<a[i][j]<<" ";
        cout<<endl;
    }
```

- //7.cpp

Exemplul de matrice este cu 5 linii, totusi 12 (=4x3) coloane! Probabil o greseala!

3. Variabilele **i** și **j** sunt de tip întreg, iar variabila **a** memorează un tablou bidimensional cu **6** linii și **6** coloane, numerotate de la **0** la **5**, având inițial toate elementele egale cu caracterul @. Fără a utiliza alte variabile, scrieți secvența de instrucțiuni de mai jos, înlocuind punctele de suspensie astfel încât, în urma executării secvenței obținute, variabila **a** să memoreze tabloul alăturat.

```
for(i=0;i<6;i++)
  for(j=0;j<6;j++)
    ..................
```

(6p.)

(	(	(	)	)	)
*	(	(	)	)	*
*	*	(	)	*	*
*	*	(	)	*	*
*	(	(	)	)	*
(	(	(	)	)	)

3. Variabilele `i` și `j` sunt de tip întreg, iar variabila `a` memorează un tablou bidimensional cu `6` linii și `6` coloane, numerotate de la `0` la `5`, având inițial toate elementele egale cu caracterul @. Fără a utiliza alte variabile, scrieți secvența de instrucțiuni de mai jos, înlocuind punctele de suspensie astfel încât, în urma executării secvenței obținute, variabila `a` să memoreze tabloul alăturat.

```
for(i=0;i<6;i++)
  for(j=0;j<6;j++)
    ..................
```

(6p.)

(	(	(	)	)	)
*	(	(	)	)	*
*	*	(	)	*	*
*	*	(	)	*	*
*	(	(	)	)	*
(	(	(	)	)	)

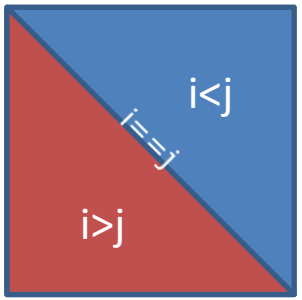
Data un element oarecare din matrice: a[i][j]

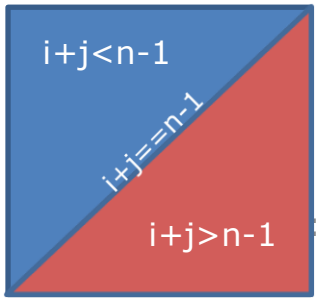
- Daca e pe diagonala principala: i == j
  - Daca este deasupra diagonalei principale: i < j
  - Daca este dedesubtul diagonalei principale: i > j

- Daca este pe diagonala secundara: i+j == n-1 (sau j = n-1-i)
  - Daca este **deasupra** diagonalei secundare: **i+j < n-1**
  - Daca este **dedesubtul** diagonalei secundare: **i+j > n-1**

Deci:

- Pe primele 3 coloane:
  - Intersectia dintre zona de SUB diag principala si cea de DEASUPRa diag secundare: *
  - Restul (
- Ultimele 3 coloane
  - Intersectia dintre zona de DEASUPRA diag principale si cea de SUB diag secundara: *
  - Restul)





3. Variabilele **i** și **j** sunt de tip întreg, iar variabila **a** memorează un tablou bidimensional cu **6** linii și **6** coloane, numerotate de la **0** la **5**, având inițial toate elementele egale cu caracterul @. Fără a utiliza alte variabile, scrieți secvența de instrucțiuni de mai jos, înlocuind punctele de suspensie astfel încât, în urma executării secvenței obținute, variabila **a** să memoreze tabloul alăturat.

```
for(i=0;i<6;i++)
   for(j=0;j<6;j++)
      ..................
```

(6p.)

(	(	(	)	)	)
*	(	(	)	)	*
*	*	(	)	*	*
*	*	(	)	*	*
*	(	(	)	)	*
(	(	(	)	)	)

Data un element oarecare din matrice: a[i][j]

- Daca e pe diagonala principala: i == j
  - Daca este deasupra diagonalei principale: i < j
  - Daca este dedesubtul diagonalei principale: i > j
- Daca este pe diagonala secundara: i+j == n-1 (sau j = n-1-i)
  - Daca este **deasupra** diagonalei secundare: **i+j < n-1**
  - Daca este **dedesubtul** diagonalei secundare: **i+j > n-1**

Deci:

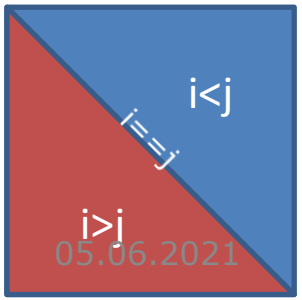
- Pe primele 3 coloane:
  - Intersectia dintre zona de SUB diag principala si cea de DEASUPRa diag secundare: *
  - Restul (
- Ultimele 3 coloane
  - Intersectia dintre zona de DEASUPRA diag principale si cea de SUB diag secundara: *
  - Restul)

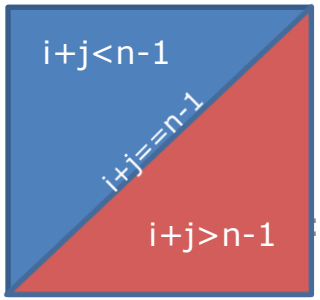
```
using namespace std;
int main(){
    int i, j;
    char a[6][6];
    for(i=0; i<6; i++)
        for(j=0; j<6; j++)
            a[i][j] = '@';

    for(i=0; i<6; i++)
        for(j=0; j<6; j++)
            if (j<=2)
                if(i>j && i+j<6-1)
                    a[i][j]='*';
                else
                    a[i][j] = '(';
            else
                if(j>i && i+j > 5)
                    a[i][j]='*';
                else
                    a[i][j] = ')';

    for(i=0;i<6;i++){
        for(j=0; j<6; j++)
            cout<<a[i][j]<<' ';
        cout<<endl;
    }

} // 11.cpp
```





Vă doresc mult succes!