Matrice pătratice

Dacă pentru un tablou bidimensional, la o anumită rulare a programului, folosim și pentru numărul de linii și pentru numărul de coloane aceeași valoare, acesta se numește pătratic (matrice pătratică). În acest material vom studia diverse particularități care apar la astfel de tablouri. În fond, tot ce am discutat la prezentarea generală a matricelor este valabil, dar aici apar câteva lucruri noi. În tabelul de mai jos, pentru o matrice pătratică de dimensiune 7 (7 linii și 7 coloane) cu liniile și coloanele indexate de la 1, am notat doar indicii elementelor.

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

Discuția din acest material se dezvoltă în jurul faptului că, așa cum am subliniat și în figură, dacă se pleacă dintr-un colț și se avansează în diagonală, se ajunge exact în colțul opus (evident că acest lucru este valabil doar pentru că matricea este pătratică).

Ceea ce este haşurat mai sus se cheamă **diagonala principală**. Dacă se pleacă din colțul dreapta al liniei de sus (1,7) și se avansează în diagonală, se va ajunge în colțul stânga jos (7,1). Zona astfel parcursă se cheamă **diagonala secundară**.

Diagonala principală și zonele delimitate de ea

Ne vom ocupa de vizitarea elementelor din anumite zone formate. Cu elementele traversate putem realiza diverse operații dintre cele clasice (sume, produse, numărări, maxime, minime, verificări etc) dar în exemplele următoare noi vom realiza, pentru simplitate, suma.

Parcurgerea elementelor de pe diagonala principală

Observăm că elementele de pe diagonala principală au indicele de linie și cel de coloană egali. Doar ele sunt elementele din matrice cu indicii egali. Așadar o primă soluție este de a traversa toată matricea și, cu un `if`, selectăm doar elementele cu această proprietate.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i==j)
            suma += a[i][j];
```

Observăm că atât pentru linii cât și pentru coloane se merge cu indicii până la `n` (am considerat o matrice pătratică de dimensiune `n`, așa cum vom considera și în exemplele următoare).

Iată și o altă variantă de a însuma elementele de pe diagonala principală:

```
for (i=1;i<=n;i++)
    suma += a[i][i];
```

Codul de mai sus este de asemenea corect, este mai scurt și este mai eficient (avem timp de ordin `n` spre deosebire de prima variantă unde timpul de executare este de ordin n^2). Cum am gândit pentru a ajunge la el?

Ne-am imaginat contorul de la `for` ca fiind linia pe care dorim să o vizităm și observăm că pe linia curentă este un singur element care ne interesează și că indicele de coloană are aceeași valoare cu acela de linie, fixat prin `for`.

O idee general valabilă este următoarea: ***dacă zona de traversat este liniară (o anume linie a matricei, o anume coloană a matricei, o anume diagonală etc) atunci vizitarea se poate face cu o singură structură repetitivă; dacă zona de traversat este ca o suprafață (matricea în sine sau o anume parte a sa dispusă pe mai multe linii și mai multe coloane), atunci vizitarea elementelor sale trebuie făcută cu două foruri, unul cu care indicăm linia pe care ne aflăm și celălalt pentru vizitarea elementelor de pe linia fixată.***

Vom vedea mai bine cele de mai sus în secțiunile următoare.

Parcurgerea elementelor din triunghiul delimitat de diagonala principală, aflate sub ea

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

Analizând tabelul cu indicii observăm că elementele aflate în zona care ne interesează au proprietatea că indicele de linie este mai mare decât cel de coloană și doar în această zonă elementele au această proprietate. Astfel, putem traversa toată matricea și, cu un `if`, selectăm aceste elemente.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i>j)
            suma += a[i][j];
```

Codul prezentat vizitează elementele din zona aflată strict sub diagonală. Pentru a include și diagonala era suficient să schimbăm doar semnul de comparare de la condiția lui if din > în >=.

O variantă mai bună decât cea prezentată este să analizăm atent cum putem vizita doar elementele din zona care ne interesează. Am reduce astfel la jumătate timpul de executare (în zona care ne interesează sunt aproximativ jumătate dintre elementele matricei).

Observăm că liniile acoperite de elementele zonei sunt între `2` și `n`.

- Pe linia `2`, coloanele sunt de la `1` la `1`
- Pe linia `3`, coloanele sunt de la `1` la `2`
- Pe linia `4`, coloanele sunt de la `1` la `3`
- ...
- Pe linia `n` coloanele sunt de la `1` la `n-1`.

Așadar, dacă vom vizita cu primul `for` liniile (`i` fiind indicele, deci linia curentă), coloana (deci indicele `j` de la al doilea `for`) va fi între `1` și `i-1`.

Iată codul:

```
for (i=1;i<=n;i++)
    for (j=1;j<i;j++)
        suma += a[i][j];
```

Această variantă are complexitatea teoretică în timp tot de ordin n^2, ca și prima, dar numărul efectiv de operații este mai mic (aproximativ jumătate).

Parcurgerea elementelor din triunghiul delimitat de diagonala principală, aflate deasupra ei.

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

Vom prezenta, ca și mai sus, două moduri de abordare:

- Observăm relația dintre indici pentru elementele zonei și le selectăm cu `if` în timpul parcurgerii întregii matrice;
- Fixăm cu un indice liniile cu elemente care ne interesează și variem cu alt indice coloana de pe linia curentă după ce observăm regula generală pe care o aplicăm fiecărei linii;

Varianta 1.

Elementele zonei dorite sunt cele cu indicele de linie strict mai mic decât cel de coloană.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i<j)
            suma += a[i][j];
```

Varianta 2.

Observăm că ne interesează linii de la `1` la `n-1`

- Pe linia `1` avem coloane de la `2` la `n`
- Pe linia `2` avem coloane de la `3` la `n`
- Pe linia `3` avem coloane de la `4` la `n`
- ...
- Pe linia `n-1` avem coloane de la `n` la `n`

```
for (i=1;i<=n;i++)
    for (j=i+1;j<=n;j++)
```

```
        suma += a[i][j];
```

În legătură cu diagonala principală și celelalte diagonale "paralele" cu ea, facem următoarele observații:

- Am spus că pentru elementele de pe diagonala principală `i=j`. Acest lucru este echivalent cu `i-j=0`.
- Elementele de pe paralela la diagonala principală aflate imediat sub ea au `i-j=1`.

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

- Elementele aflate pe diagonala aflate sub cea anterioară au `i-j=2`.
- În cealaltă parte, elementele diagonalei aflată imediat deasupra celei principale au `i-j=-1`.

În colcluzie, elementele aflate pe diagonala principală sau pe aceeași paralelă la ea au diferența indicilor constantă.

Putem folosi rezultatul de mai sus pentru a gândi rapid un cod optim de a traversa elementele aflate pe o astfel de diagonală. În continuare calculăm suma elementelor aflate imediat sub diagonala principală.

```
for (i=2;i<=n;i++)
    suma += a[i][i-1];
```

Așa cum observasem anterior, fiecare astfel de diagonală căutăm să o parcurgem cu un singur `for` întucât ea arată liniar.

Diagonala secundară și zonele delimitate de ea

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

De data aceasta vom porni cumva invers cu observațiile și anume: pentru elementele de pe diagonala secundară suma indicilor este `n+1`. Uitându-ne atent se vede că pe fiecare paralelă la diagonala secundară

suma indicilor este constantă. Deci pe paralela aflată imediat deasupra suma indicilor este `n`, pe cea aflată încă un nivel mai sus suma indicilor este `n-1` etc.

Deci, pe paralelele la diagonala principală avem diferența indicilor constantă iar pe paralelele la diagonala secundară avem suma indicilor constantă.

Parcurgerea elementelor de pe diagonala secundară

Varianta 1. Este cea neoptimă: parcurgem toată matricea și punem condiția `i+j=n+1`

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i+j == n+1)
            suma += a[i][j];
```

Varianta 2. Observăm că zona de parcurs este liniară deci trebuie să putem rezolva cu un `for`. Fixăm indicele acestuia ca fiind linia unde avem elemente (de la `1` la `n` aici) și coloana o calculăm ca find `n+1-liniaCurenta`.

```
for (i=1;i<=n;i++)
    suma += a[i][n+1-i];  /// sau a[i][n-i+1] cum intalnim des
```

Parcurgerea elementelor din triunghiul delimitat de diagonala secundară, aflate sub ea

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

Prima variantă se bazează pe observația că elementele acestei zone sunt cele pentru cate `i+j>n+1`. Așadar putem parcurge toată matricea și selectăm cu `if` doar aceste elemente.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i+j > n+1)
            suma += a[i][j];
```

Varianta a doua, cea în care vizităm exact elementele zonei se bazează pe observația:

- Avem elemente pe linii de la `2` la `n`;

- La o anume linie `i`, elementele care ne interesează sunt cele aflate după cel de pe diagonala secundară de pe linia `i` și până la finalul liniei. Așadar, indicii sunt de la `(n+1-i)+1` până la `n`. Am scris în scop didactic formula dar în cod scriem compact `n-i+2`.

```
for (i=2;i<=n;i++)
    for (j=n-i+2; j<=n; j++)
        suma += a[i][j];
```

Parcurgerea elementelor din triunghiul delimitat de diagonala secundară, aflate deasupra ei

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

Varianta 1: elementele care ne interesează sunt cele cu suma indicilor mai mică decât `n+1`.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i+j < n+1)
            suma += a[i][j];
```

Varianta 2: facem observațiile:

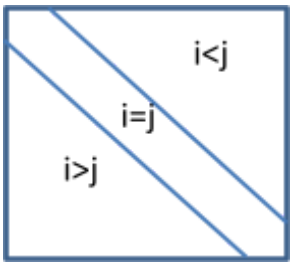
- Avem elemente pe linii de la `1` la `n-1`;
- La o anume linie `i`, elementele care ne interesează sunt cele aflate de la începutul liniei, deci coloana `1`, până înainte de cel de pe diagonala secondară, adică mergem cu coloana până la `n-i`.

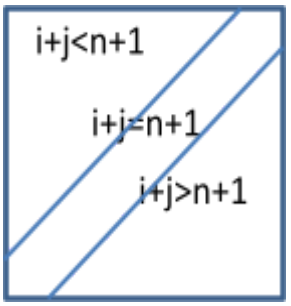
```
for (i=1;i<n;i++)
    for (j=1; j<=n-i; j++)
        suma += a[i][j];
```

În rezumat, relațiile dintre indici pentru diagonalele matricei pătratice și pentru zonele delimitate de ele sunt:

Diagonala principală și zonele

Diagonala secundară și zonele





Parcurgerea zonelor delimitate de câte două diagonale

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

În figura de mai sus am haşurat elementele de pe cele două diagonale dar şi pe cele din zona de sus delimitată de ele. Ca şi la cele prezentate mai sus, avem două abordări pentru a le vizita.

Prima este să identificăm condiţiile pe care le îndeplinesc indicii doar pentru elemente din acea zonă. Aici observăm că ele sunt atât deasupra diagonalei principale (`i<j`) cât şi deasupra diagonalei secundare (`i+j<n+1`). Aşadar putem traversa toată matricea şi să punem simultan cele două condiţii.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i < j && i+j < n+1)
            suma += a[i][j];
```

Varianta a doua este să încercăm traversarea doar a elementelor care ne interesează. Fiind vorba de o suprafaţă ne gândim că sunt necesare două foruri, deci tot n^2 va fi complexitatea însă practic noi reducem de `4` ori timpul de calcul (observăm că elementele zonei reprezintă aproximativ un sfert din toată matricea).

Liniile pe care se află elementele noastre sunt începând cu `1` şi până la jumătate. Formula după care obţinem linia de jos depinde şi de paritatea lui `n`. Astfel, dacă `n` este impar este suficient `n/2` (câtul), iar dacă `n` este par trebuie `n/2-1` (elementele de pe linia `n/2` sunt şi pe diagonale şi noi avem haşurată o zonă fără diagonale). Putem combina cele două într-o singură formulă care nu depinde de paritatea lui `n`, şi anume: `(n-1)/2`.

Mai observăm că pe o linie `i` fixată elementele care ne interesează se află după cel de pe diagonala principală şi până înaintea celui de pe diagonala secundară. Astfel obţinem codul:

```
for (i=1;i<=(n-1)/2;i++)
    for (j=i+1;j<n+1-i;j++)
        suma += a[i][j];
```

Probleme rezolvate

Ideile folosite la rezolvarea acestor probleme sunt unele clasice, des întâlnite în practică.

1. Se citește de la tastatură un număr n. Să se construiască o matrice pătratică de dimensiune n, după următoarele reguli:

- Elementele de pe diagonala principală să aibă valoarea 1;
- Elementele din triunghiul de sub diagonala principală să aibă valoarea 2;
- Elementele din triunghiul de deasupra diagonalei principale să aibă valoarea 3.

Exemplu:

Date de intrare	Date de ieșire
5	1 3 3 3 3 2 1 3 3 3 2 2 1 3 3 2 2 2 1 3 2 2 2 2 1

Rezolvare

Observăm că de data aceasta nu se mai citesc de la tastatură toate valorile unei matrice. Avem de construit una pornind de la dimensiunea sa. Întrucât avem de vizitat toate elementele se impune folosirea metodei care traversează toată matricea și care, cu `if`, identifică zona în care ne aflăm. Mai departe prezentăm doar secvența de cod presupunând că matricea de construit se numește `a`.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++) {
        if (i == j)
            a[i][j] = 1;
        else
            if (i > j)
                a[i][j] = 2;
            else
                a[i][j] = 3;
    }
```

Am prezentat doar secvența de memorare de valori în matrice fără a mai afișa ulterior tabloul.

2. Se citește de la tastatură un număr `n`. Să se construiască o matrice pătratică de dimensiune `n` după regulile:

- Elementele celor două diagonale au valoarea `1`;
- Elementele triunghiului de sus delimitat de cele două diagonale au valoarea `2`;
- Elementele triunghiului de jos delimitat de cele două diagonale au valoarea `3`;
- Elementele triunghiului din stânga delimitat de cele două diagonale au valoarea `4`;
- Elementele triunghiului din dreapta delimitat de cele două diagonale au valoarea `5`;

Exemplu:

Date de intrare	Date de ieșire
5	1 2 2 2 1 4 1 2 1 5 4 4 1 5 5 4 1 3 1 5 1 3 3 3 1

Rezolvare

Aplicăm aceeași tehnică de la problema anterioară, parcurgând toată matricea și identificând zona în care ne aflăm. Pentru zonele triunghiulare delimitate de două diagonale folosim `if` cu două condiții, câte o condiție pentru a indica partea de care se află elementul curent față de fiecare diagonală.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++) {
        if (i == j || i+j == n+1)
            a[i][j] = 1;
        if (i < j || i+j < n+1)
            a[i][j] = 2;
        if (i > j || i+j > n+1)
            a[i][j] = 3;
        if (i > j || i+j < n+1)
            a[i][j] = 4;
        if (i < j || i+j > n+1)
            a[i][j] = 4;
    }
```

La această problemă am ales varianta cu `if`-uri succesive în locul celei cu `if-else-if` întucât codul este mai compact.

3. Se citește de la tastatură un număr `n`. Să se construiască o matrice pătratică după următoarele reguli:

Elementele de pe prima linie, de pe prima coloană și de pe ultima coloană sunt `1`. Fiecare dintre celelalte elemente se obține ca sumă a celor trei vecine aflate pe linia de deasupra sa: cel de sus, cel din stânga, cel din dreapta.

Exemplu:

Date de intrare	Date de ieșire
5	1 1 1 1 1 1 3 3 3 1 1 7 9 7 1 1 17 23 17 1 1 41 57 41 1

Rezolvare

Inițializăm separat elementele care trebuie să fie `1` apoi parcurgem restul elementelor și pentru cel curent realizăm atribuirea `a[i][j]=a[i-1][j-1]+a[i-1][j]+a[i-1][j+1]`.

```
for (i=1;i<=n;i++)
    a[i][1] = a[i][n] = a[1][i] = 1;
for (i=2;i<=n;i++)
    for (j=2;j<=n-1;j++)
        a[i][j] = a[i-1][j-1] + a[i-1][j] + a[j-1][j+1];
```

O altă soluție este să parcurgem toată matricea și să identificăm, cu `if`, locul în care ne aflăm.

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
        if (i==1 || j==1 || j==n)
```

```
            a[i][j] = 1;
        else
            a[i][j] = a[i-1][j-1] + a[i-1][j] + a[j-1][j+1];
```

4. Se citește de la tastatură `n`. Să se construiască o matrice pătratică de dimensiune `n` după următoarele reguli:

- Elementele de pe ultima linie sunt primele `n` numere naturale în ordine crescătoare.
- Fiecare dintre celelalte elemente se calculează ca dublul valorii elementului aflat sub el).

Pentru n=6 matricea s-ar obține:

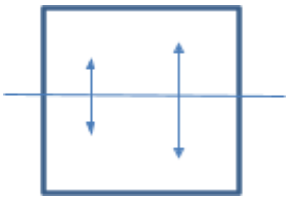
32	64	96	128	160	192
16	32	48	64	80	96
8	16	24	32	40	48
4	8	12	16	20	24
2	4	6	8	10	12
1	2	3	4	5	6

Rezolvare

În acest caz se inițializează doar ultima linie apoi, pentru celelalte linii, matricea se parcurge de jos în sus (este necesar așa pentru a avea calculat deja elementul de care avem nevoie – cel de sub).

```
for (i=1;i<=n;i++)
    a[n][i] = i;
for (i=n-1;i>=1;i--)
    for (j=1;j<=n;j++)
        a[i][j] = 2*a[i+1][j];
```

5. Dată fiind o matrice pătratică de dimensiune `n`, să se oglindească față de linia din mijloc. Fiecare element va fi împerecheat cu altul ca în figura de mai jos și valorile lor se vor interschimba.



Exemplu:

Date de intrare	Date de ieșire
5 1 2 3 4 5 1 0 1 0 1 9 8 7 6 5 5 4 3 2 1 1 1 1 1 1	1 1 1 1 1 5 4 3 2 1 9 8 7 6 5 1 0 1 0 1 1 2 3 4 5

Rezolvare

Din analiza tabelului cu indici rezultă că elementul `a[i][j]` va trebui interschimbat cu elementul `a[n+1-i][j]`. Acest lucru se mai poate deduce și observând că elementele care se interschimbă rămân pe aceeași coloană iar liniile lor sunt simetrice.

Așadar vom parcurge doar liniile din jumătatea de sus, realizând pentru elementul curent o operație de interschimbare cu perechea lui.

```
for (i=1;i<=n/2;i++)
    for (j=1;j<=n;j++) {
        aux = a[i][j];
        a[i][j] = a[n+1-i][j];
        a[n+1-i][j] = aux;
    }
```

6. Dată fiind o matrice pătratică de dimensiune `n`, să se verifice dacă valorile sale sunt dispuse simetric față de doagonala principală.

Rezolvare

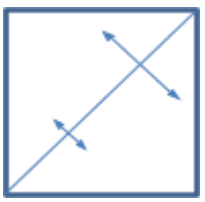
Analizând tabelul cu indici, observăm că elementele dispuse simetric față de diagonala principală sunt de forma `a[i][j]` și `a[j][i]`.

```
ok = 1;
for (i=2;i<=n;i++)
    for (j=1;j<i;j++)
        if (a[i][j] != a[j][i])
            ok = 0;
if (ok)
    cout<<"da";
else
    cout<<"nu";
```

Observăm că a fost necesată fixarea cu `i,j` doar pentru elementele aflate de o parte a diagonalei (comparând elementul fixat cu simetricul său față de diagonală).

7. Dată fiind o matrice pătratică de dimensiune `n`, să se calculeze câte elemente sunt egale cu simetricul față de diagonala secundară.

Rezolvare



Ca de obicei se cuvine analiza tabelului cu indici prezentat la începutul capitolului. Se observă, ca și în cazul diagonalei principale că liniile se duc în coloane și coloanele în linii, adică între element și simetricul său, `i` și `j` se inversează. Mai apare însă în plus faptul că fiecare dintre indici se și simetrizează față de `n`. Așadar, pentru elementul fixat `a[i][j]`, simetricul său față de diagonala secundară este `a[n+1-j][n+1-i]`. Vom parcurge cu `i,j` doar unul dintre triunghiurile delimitate de diagonala secundară.

```
for (i=1;i<n;i++)
```

```
    for (j=1;i+j<n+1;j++)
        if (a[i][j] == a[n+1-j][n+1-i])
            sol++;
```

8. Dată fiind o matrice pătratică de dimensiune `n` (`n`-impar), să se numere câte perechi de elemente poziționate simetric față de elementul din mijloc sunt egale.

Rezolvare

Aici se observă că elementele de pe o linie au corespondente elementele de pe linia simetrică și la fel pentru coloane. Așadar, față de problema anterioară nu se mai schimbă liniile și coloanele, deci elementul de pe poziția `i,j` are ca și corespondent pe cel de pe poziția `n+1-i,n+1-j`. De remarcat faptul că este valabil codul prezentat și pentru cazul cu n par, dar în acest caz nu există efectiv un element în mijlocul matricei (centrul de simetrie este deci un punct imaginar în mijloc).

```
for (i=1;i<n;i++)
    for (j=1;i+j<n+1;j++)
        if (a[i][j] == a[n+1-i][n+1-j])
            sol++;
```

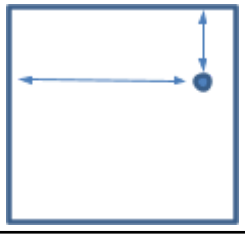
9. Dată fiind o matrice pătratică de ordin `n`, să se "răstoarne" aceasta cu `90` de grade la stânga, ca în exemplu (observați că am notat în fiecare celulă indicii ei).

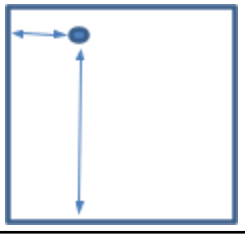
1,1	1,2	1,3	1,4	1,5	1,6
2,1	2,2	2,3	2,4	2,5	2,6
3,1	3,2	3,3	3,4	3,5	3,6
4,1	4,2	4,3	4,4	4,5	4,6
5,1	5,2	5,3	5,4	5,5	5,6
6,1	6,2	6,3	6,4	6,5	6,6

1,6	2,6	3,6	4,6	5,6	6,6
1,5	2,5	3,5	4,5	5,5	6,5
1,4	2,4	3,4	4,4	5,4	6,4
1,3	2,3	3,3	4,3	5,3	6,3
1,2	2,2	3,2	4,2	5,2	6,2
1,1	2,1	3,1	4,1	5,1	6,1

Rezolvare

Ne putem imagina un element de mutat și poziția lui, apoi locul unde trebuie să ajungă.



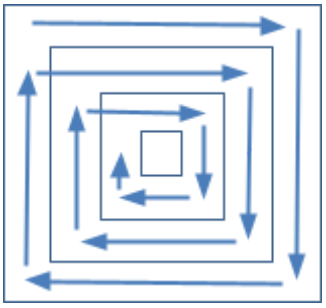


Observația esențială, generală, pentru rotirile cu 90 de grade este că liniile devin coloane și coloanele devin linii dar, spre deosebire de simetrizarea față de diagonale, unde acest lucru se întâmplă la fel, acum exact una dintre dimensiuni se și simetrizează. Atfel spus, fie i,j trece în n+1j,i fie i,j trece în j,n+1-i.

La o analiză atentă observăm că în cazul acestor rotații trebuie parcursă întreaga matrice și că este nevoie de o a doua matrice în care să se scrie elementele primeia în ordinea dorită, în caz contrar apărând suprascrieri.

```
for (i=1;i<=n;i++)
    for (j=1;i<=n;j++)
        b[n+1-j][i] = a[i][j];
```

10. Se dă un număr n și elementele întregi ale unei matrice pătratice de ordin n. Să se afișeze elementele matricei în ordinea dată de o parcurgere în spirală care începe din colțul stânga sus, ca în desenul de mai jos.

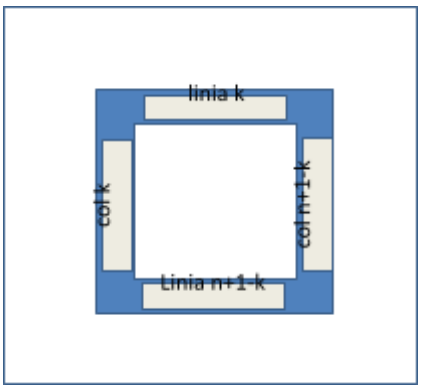


De exemplu, dacă matricea este de dimensiune 6 și conține elementele:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Pe ecran trebuie afișat: 1 2 3 4 5 6 12 18 24 30 36 35 34 33 32 31 25 19 13 7 8 9 10 11 17 23 29 28 27 26 20 14 15 16 22 21.

Observăm că mai întâi se afișează elementele primului contur, începând din colțul stânga sus al său (în ordine: linia 1, coloana n, linia n, coloana 1), apoi se afișează elementele celui de-al doilea contur, de asemenea începând din colțul stânga sus al său (linia 2, coloana n-1, linia n-1, coloana 2) etc. Sunt n/2 astfel de contururi iar dacă n este impar mai avem de afișat la final elementul din mijloc. Așadar ne vom concentra pe modul în care afișăm un anume contur k, unde k este cuprins între 1 și n/2.



Așadar, pentru a parcurge conturul `k` primul pas este să parcurgem de la stânga la dreapta elementele sale aflate pe linia `k`. Acestea se observă că sunt cele de pe coloane de la `k` la `n+1-k`.

```
for (i=k;i<=n+1-k;i++)
    cout<<a[k][i]<<" ";
```

Apoi se afișează elementele de pe coloana din dreapta, `n+1-k`. Acestea se afișează de sus în jos iar liniile pe care se află sunt de la `k+1` la `n+1-k` (elementul de pe linia `k` îl sărim acum pentru că l-am afișat deja la prima parcurgere).

```
for (i=k+1;i<=n+1-k;i++)
    cout<<a[i][n+1-k]<<" ";
```

Al treilea pas este să afișăm elementele de pe latura de jos a conturului. Acestea sunt pe linia `n+1-k` a matricei, aflate pe coloane de la `n+1-k-1` până la `k` (le-am scris în ordinea parcurgerii, de la dreapta la stânga și am sărit elementul din dreapta, care a fost afișat la parcurgerea anterioară).

```
for (i=n-k;i>=k;i--)
    cout<<a[n+1-k][i]<<" ";
```

În fine, elementele de pe latura stângă a conturului, de jos în sus, adică acelea aflate pe coloana `k` a matricei și pe linii de la `n-k` la `k+1` (acum am ocolit ambele colțuri).

```
for (i=n-k;i>=k+1;i--)
    cout<<a[i][k]<<" ";
```

În concluzie, scriem cele de mai sus pentru fiecare `k` de la `1` la `n/2` și obținem:

```
for (k=1;k<=n/2;k++) {
    for (i=k;i<=n+1-k;i++)
        cout<<a[k][i]<<" ";
    for (i=k+1;i<=n+1-k;i++)
        cout<<a[i][n+1-k]<<" ";
    for (i=n-k;i>=k;i--)
        cout<<a[n+1-k][i]<<" ";
    for (i=n-k;i>=k+1;i--)
        cout<<a[i][k]<<" ";
}
if (n%2 != 0)
    cout<<a[n/2+1][n/2+1];
```

11. Se dă o matrice pătratică cu `n` linii și `n` coloane și elemente numere naturale mai mici decât `1000`. Să se afișeze în ordine strict crescătoare valorile situate sub diagonala principală și deasupra diagonalei secundare. Dacă o valoare apare în zona respectivă de mai multe ori, se va afișa o singură dată (pbinfo.ro, #729).

Exemplu:

Date de intrare	Date de ieșire
6 10 8 5 8 4 2 6 5 3 1 3 8 8 1 4 7 8 8 5 1 9 6 6 1 8 9 10 1 3 6 8 2 3 3 9 6	1 5 6 8

Rezolvare

Ne folosim de faptul că elementele sunt naturale de maxim trei cifre și vom folosi un vector de frecvență în care marcăm valorile care apar. Parcurgerea indicilor acestuia în ordine crescătoare și afișarea acelora unde am marcat ne oferă rezultatul.

```
#include <iostream>
#include <algorithm>
using namespace std;
int n, a[201][201], f[1001], i, j;
int main (){
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(i=1; i<=n; i++){
        for (j=1;j<=n;j++)
            if (i>j && i+j<n+1) {
                f[a[i][j]] = 1;
            }
    }
    for (i=0; i<=1000; i++)
        if (f[i] == 1)
            cout<<i<<" ";
    return 0;
}
```

De remarcat și faptul că era suficient să folosim vectorul de frecvență, declararea matricei nefiind obligatorie întrucât puteam utiliza aceeași variabilă în care să citim toate elementele, pe rând.