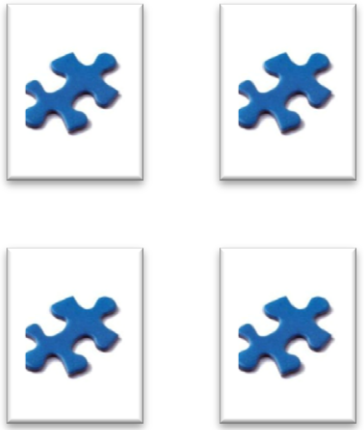
Tipuri structurate de date

Tipul tablou



Sumar

Competenţe generale

- *implementarea algoritmilor într-un limbaj de programare*
- *aplicarea algoritmilor fundamentali în prelucrarea datelor*

Competenţe specifice

- *identificarea necesităţii structurării datelor în tablouri*
- *prelucrarea datelor structurate*
- *utilizarea unui mediu de programare C++*

O ***structură de date*** reprezintă un ansamblu (o colecţie) de date organizate după anumite reguli, care depind de tipul de structură.

Dintre tipurile de structuri de date fac parte:

- tablourile de memorie;
- fişiere.

Tablouri

Un ***tablou*** este o structură format dintr-un număr fix de componente de acelaşi tip, numit tip de bază.

Tabloul reprezintă o zonă de memorie cărei i se atribuie un nume şi care permite memorarea mai multor date de acelaşi tip. Aceste date pot fi tratate ca un tot unitar sau ca date elementare independente.

Tablourile sunt de două tipuri:

- tablouri unidimensionale (vectori);
- tablouri bidimensionale (matrici).

Tablouri unidimensionale (vectori)

Pentru a prelucra un set de valori de acelaşi tip, acesta trebuie memorat într-o structură de date. O astfel de structură de date se numeşte ***şir***, iar valorile respective se numesc ***elementele şirului***.

În limbajul C++ elementele unui şir se memorează într-o singură variabilă indexată numită ***tablou unidimensional*** sau ***vector***.

Elementele şirului memorate într-un vector se numesc ***elementele vectorului*** sau ***componentele vectorului***.

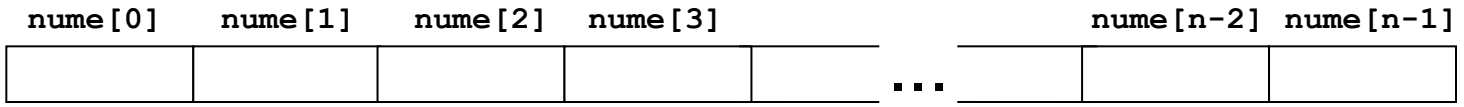
Declarare vector

Sintaxa:

```
tip_dată nume[nr_elemente];
```

unde:

- **`tip_dată`** precizează tipul elementelor vectorului;
- **`nume`** este identificatorul vectorului;
- **`nr_elemente`** este o constantă întreagă care reprezintă numărul de elemente ale vectorului.



Fiecare element este indicat prin locul pe care îl ocupă în cadrul vectorului. Poziţiile elementelor în vector sunt numerotate succesiv începând cu **0**.

Exemple

1. Se declară un vector **a** cu **10** elemente de tip **int**:

```
int a[10];
```

2. Se declară un vector **x** cu **5** elemente de tip **float**:

```
float x[5];
```

3. Se declară un vector **c** cu **8** elemente de tip **char**:

```
char c[8];
```

4. Iniţializarea elementelor unui vector la declarare:

```
int v[]={21, 4, 361, 55};
```

5. Declararea unei constante întregi care va fi folosită la declararea vectorului:

```
const int NR_ELEMENTE=50;
int v[NR_ELEMENTE];
```

Referirea la un element al vectorului se face prin construcţia:

`nume[indice]`

unde:

- **`nume`** este identificatorul vectorului;
- **`indice`** este numărul de ordine al elementului în cadrul vectorului.

Exemplu

```
int v[4];
```

	V[0]	V[1]	V[2]	V[3]
	18	-4	133	75

sau

v	18	-4	133	75
	0	1	2	3

Citire vector

- se citeşte mai întâi numărul de elemente ale vectorului;
- se citesc pe rând, unul câte unul elementele vectorului;

```
cout<<"Numarul de elemente:";
cin>>n;
for(i=1;i<=n;i++)
{
      cout<<"v["<<i<<"]=";
      cin>>v[i];
}
```

Afişare vector

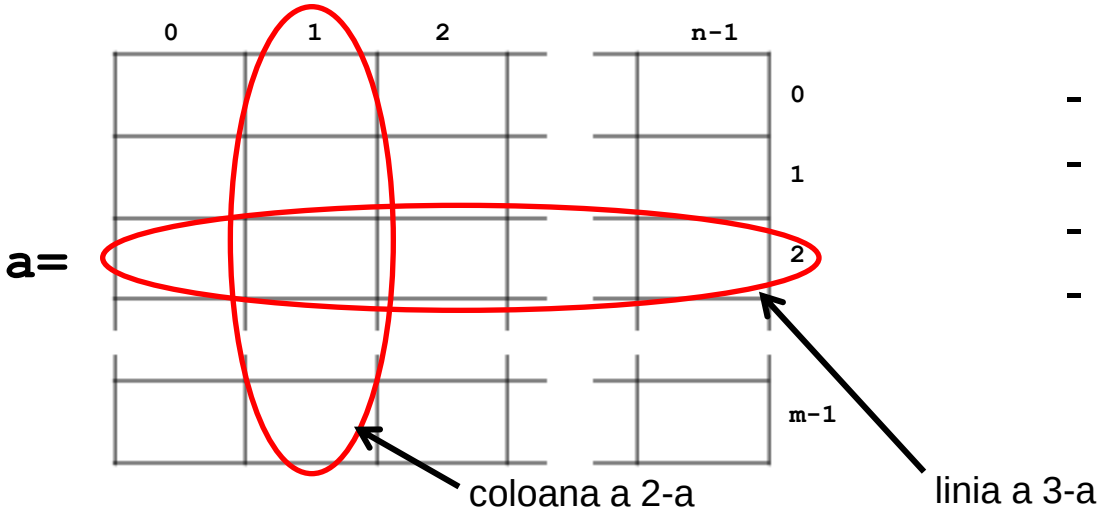
- se realizează afişând pe rând elementele vectorului;

```
for(i=1;i<=n;i++)
    cout<<v[i]<<' ';
```

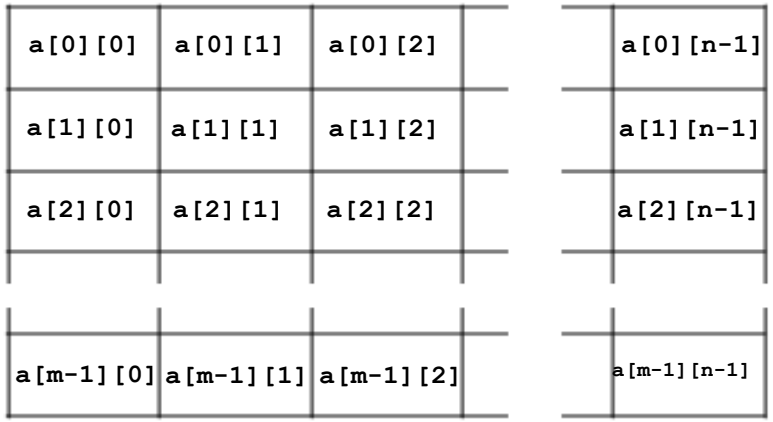
Tablouri bidimensionale (matrici)

Un tablou bidimensional (matrice) este un tablou cu elemente de acelaşi tip, dispuse pe linii şi coloane. Fiecare element al matricii se află pe o anumită linie şi pe o anumită coloană.

Considerăm o matrice notată **a** cu **m** linii şi **n** coloane:



- ***m*** numărul de linii
- ***n*** numărul de coloane
- ***i*** indicele liniei
- ***j*** indicele coloanei



Declarare matrice

Sintaxa:

```
tip_dată nume[nr_1][nr_2];
```

unde:

- **tip_dată** precizează tipul elementelor matricei;
- **nume** este identificatorul matricei:
- **nr_1** şi **nr_2** două constante întregi care specifică numărul de elemente ale matricei pentru fiecare dimensiune, astfel:
  - **nr_1** numărul de linii;
  - **nr_2** numărul de coloane.

Exemple

1. Se declară o matrice **a** cu **100** elemente de tip **int**:

```
int a[10][10];
```

2. Se declară o matrice **x** cu **40** elemente de tip **float**:

```
float x[5][8];
```

3. Se declară o matrice **c** cu **8** elemente de tip **char**:

```
char c[4][2];
```

4. Iniţializarea elementelor unei matrici la declarare:

```
int a[2][4]={1,2,3,4,5,6,7,8};
```

5. Declararea unei constante întregi care va fi folosită la declararea matricii:

```
const int NR_ELEMENTE=50;
int x[NR_ELEMENTE][NR_ELEMENTE];
```

Referirea la un element al matricei se face prin construcţia:

```
nume[indice_1][indice_2]
```

sau

```
nume[indice_2][indice_1]
```

unde:

- **nume** este identificatorul matricei;
- **indice_1** este numărul liniei pe care se află elementul în matrice;
- **indice_2** este numărul coloanei pe care se află elementul în matricei.

Matrice pătratică

În cazul în care numărul de linii este identic cu numărul de coloane, matricea se numeşte ***pătratică***.

Exemplu

```
n=4
int a[5][5];
```

n - numărul de linii şi de coloane
nxn – numărul de elemente ale matricei

a=

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

Citire matrice

- se citesc mai întâi numărul de linii şi numărul de coloane ale matricei;
- se citesc pe rând, unul câte unul elementele matricei, de la stânga la dreapta şi de sus în jos:

```
cout<<"Numarul de linii:";
cin>>m;
cout<<"Numarul de coloane:";
cin>>n;
for(i=1;i<=m;i++)
   for(j=1;j<=n;j++)
   {
      cout<<"a["<<i<<"]["<<j<<"]=";
      cin>>a[i][j];
   }
```

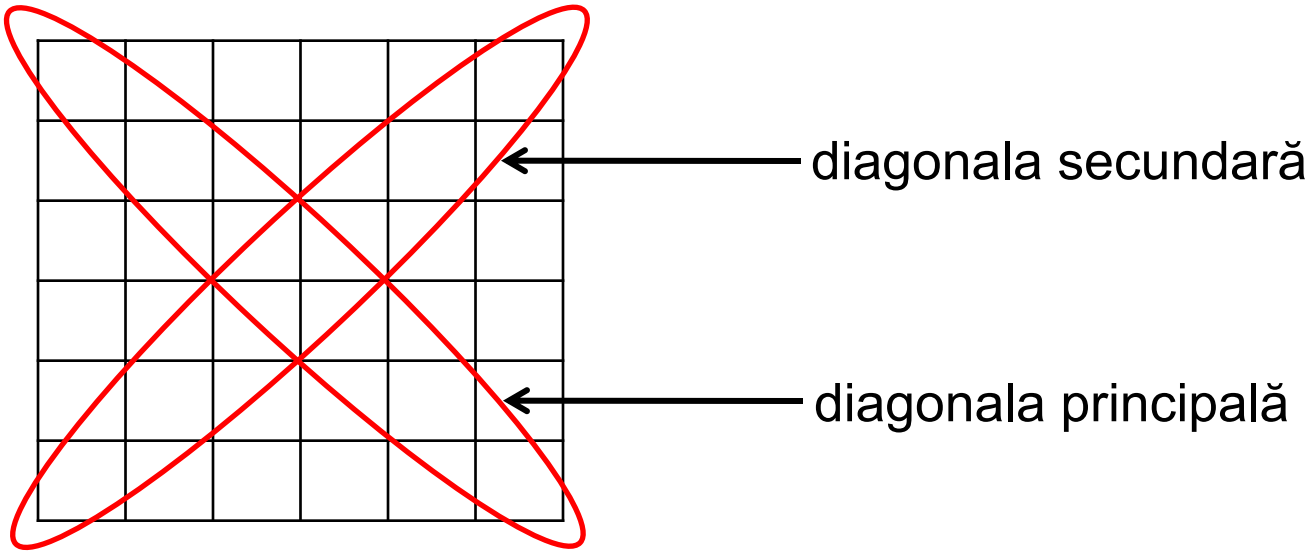
Afişare matrice

- se realizează afişând pe rând elementele matricei, linie cu linie;

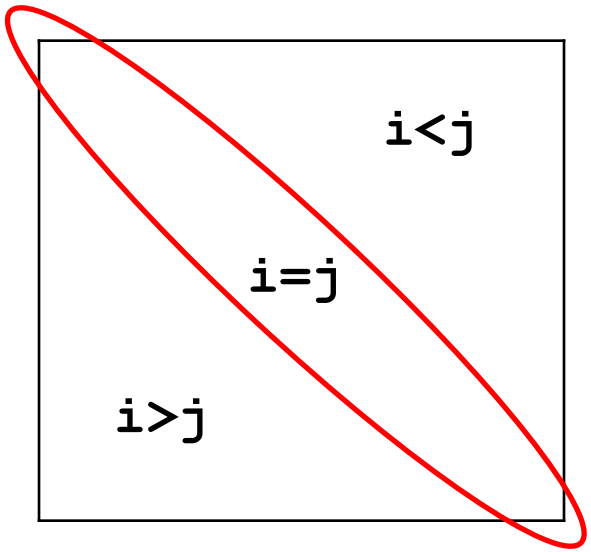
```
for(i=1;i<=m;i++)
{
    for(j=1;j<=n;j++)
       cout<<a[i][j]<<' ';
    cout<<endl;
}
```

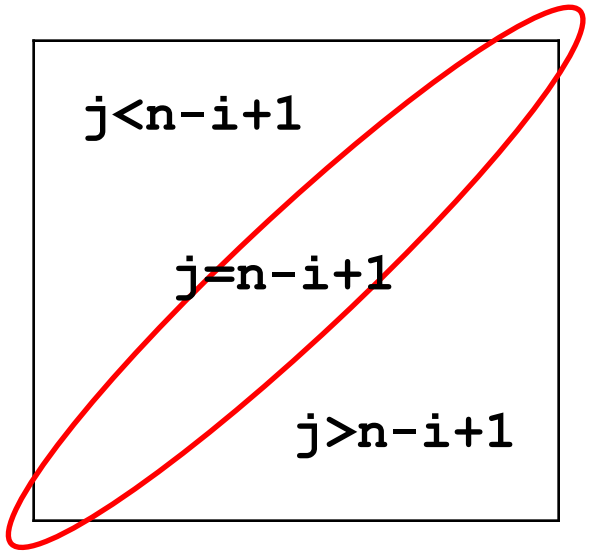
Diagonalele matricei

În cazul unei matrici pătratice, se pot prelucra elementele matricei în funcţie de diagonalele acesteia.



Parcurgerea elementelor unei matrici în funcţie de diagonale





Observaţie: `i∈[1,n]`, `j∈[1,n]`

Fişe de lucru

- Aplicaţii tablouri unidimensionale (vectori)
- Aplicaţii tablouri bidimensionale (matrici)

6. Bibliografie şi webografie

1. Miloşescu Mariana, *Informatică. Manual pentru clasa a IX-a*, Editura Didactică şi Pedagogică, Bucureşti, 2004
2. Munteanu Florin, *Programarea calculatoarelor. Manual pentru licee de informatică clasele X-XII*, Editura Didactică şi Pedagogică, Bucureşti, 1994
3. Logofătu Doina, *Bazele programării în C++*, Editura Polirom, Iaşi, 2006
4. Popescu C., *Culegere de probleme de informatică*, Editura Donaris-Info, Sibiu, 2002
5. Ministerul Educaţiei, Cercetării şi Tineretului, Centrul Naţional pentru Curriculum şi Evaluare în Învăţământul Preuniversitar, *Proba scrisă la informatică. Examenul de bacalaureat – Variante (1-100)* , Bucureşti 2008
6. http://en.wikipedia.org/wiki/Vector_(C%2B%2B)
7. http://www.tutorialeprogramare.ro/Tutorial%20C.html
8. http://ro.wikipedia.org/wiki/Matrice_(matematic%C4%83)