Cursul nr. 3

STRUCTURI DE DATE

Matrice

(Tablou bidimensional)

Lector dr. Dorin IORDACHE

Agenda

01

Declarare matrice

02

Implementare

03

Operatii

04

Exemple

DECLARARE
Matrice

01

Matrice (tablou bidimensional)

O listă de valori cu același tip de date care sunt stocate folosind un singur nume de grup.

Declarație generală a matricei:

tipul de date nume-matrice[număr-de-articole1] [număr-de-articole2];

Numărul de elemente trebuie specificat înainte de declarare:

float matr[10][20];

Matrice (Tablou bidimensional)

- **bidimensional**

Matrice

Elementele individuale ale matricei pot fi accesate prin specificarea numelui acestuia și a indexului liniei si al coloanei elementului:

matrice[3][2];

Atenție:
indicii au valori de la 0 la numărul de elemente -1!!

Matrice (tablou multi-dimensional)

Forma generală de declarare a tablourilor N-dimensional este:

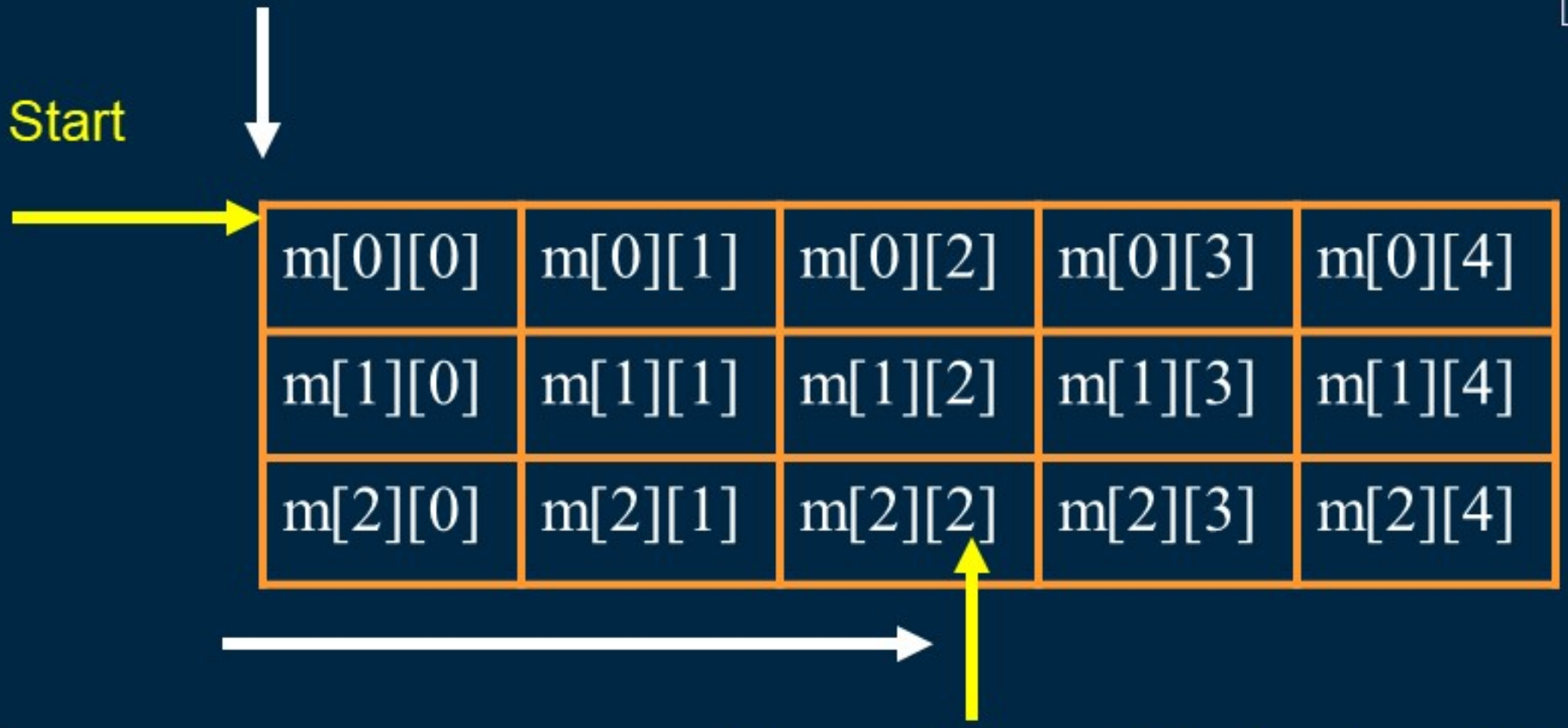
tipData numeMatrice[dimensiune1][dimensiune2]....[dimensiuneN];

tipData: Tipul de date care urmează să fie stocate în matrice.

numeMatrice: numele variabilei

dimensiune1, dimensiune2,... ,dimensiuneN: Dimensiunile vectorului

Numele tabloului m identifică locația de pornire a acestuia



Inițializarea matricei

Matricea poate fi inițializata în timpul declarației

```
int m[3][4] = {0, 1 ,2 ,3 ,4 , 5 , 6 , 7 , 8 , 9 , 10 , 11};

 int x[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}};
```

Care este diferența dintre următoarele două declarații?

sau

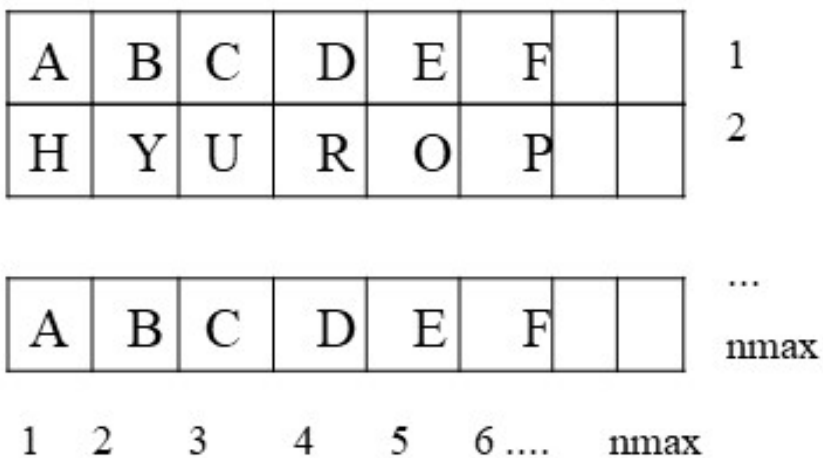
```
integer m[3,4]
for i -> 1,3 do
        for j->1,4 do
                read m[I,j]
        enfor
endfor
```

IMPLEMENTARE

02

Implementarea matricelor

Tablou



^

adresa unui element =
adresa de început + deplasament

Obs:
deplasament = 0 pt primul element
= 1 pt al doilea
etc.

Implementarea structurii liniare

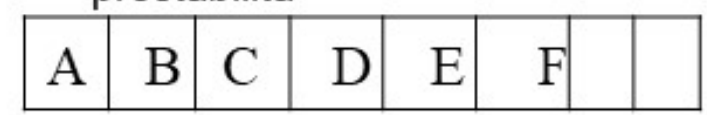
Tablou

Avantaje:

- Acces rapid pe baza indexului
- Se stochează doar valorile elementelor

Dezavantaje:

- Dimensiunea maximă este prestabilită



adresa de început

adresa unui element =
adresa de început + deplasament
(corelat cu valoarea indexului)
Obs:
deplasament = 0 pt primul element
= 1 pt al doilea
etc.

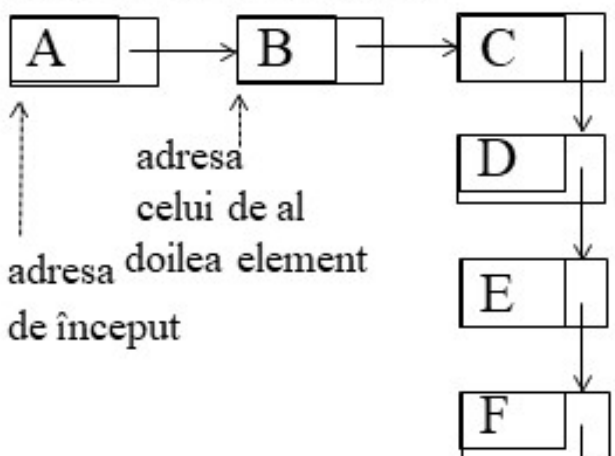
Structură (listă) înlănțuită

Avantaje:

- Dimensiune flexibilă
- Cost mic la inserare/ eliminare

Dezavantaje:

- Necesită stocarea unor informații adiționale (ex: adresa elem. următor)
- Accesul aleator nu este facil



Implementare utilizând tablouri

Matrice: x[1..Nmax][1…Lmax] - zona maximă alocată pentru stocarea structurii NxL – numărul efectiv de elemente

Complexitatea operațiilor:

- Interogare după poziție
  - Elementul aflat pe o anumită poziție:

    s.x[l,j] cost:$\Theta(1)$
  - Elementul următor/ anterior unui element specificat (element curent):

    s.x[i-1,j] sau s.x[i+1,j] cost: $\Theta(1)$

Implementare utilizând tablouri

Tablou: x[1..nmax] - zona maximă alocată pentru stocarea structurii
n – numărul efectiv de elemente

Structura: s are două componente (câmpuri):
s.x (tabloul ce conține valorile)
s.n (numărul elementelor)

Complexitatea operațiilor: Interogare după valoare

- Elementul care conține o valoare specificată
  căutare secvențială cost:O(n)
- Elementul care conține cea mai mică/ mare valoare
  determinare minim/maxim cost:$\Theta(n)$
- Elementul care conține o valoare specificată prin poziția relativă în raport cu alte valori (ex: al treilea element în ordine crescătoare, elementul median etc)
  selecția celui de al k-lea element în ordine crescătoare/ descrescătoare:
  - varianta bazată pe sortare parțială prin metoda selecției: $\Theta(kn)$
  - varianta bazată pe ideea de la quicksort : O(n) în medie

OPERATII

03

Operatii

Parcurgerea elementelor

Fie A o matrice cu M linii si N coloane, Acest algoritm parcurge matricea A și aplică operația PROCESS fiecărui element al acesteia.

```
1.    Repeat For I = 1 to M
2.          Repeat For J = 1 to N
3.                Apply PROCESS to A[I][J]
          [End of Step 2 For Loop]
     [End of Step 1 For Loop]
4.    Exit
```

Operatii

Transpusa unei matrice A in B

```
1.    Repeat For I = 1 to M
2.          Repeat For J = 1 to N
3.                Set B[J][I] = A[I][J]
          [End of Step 2 For Loop]
     [End of Step 1 For Loop]
4.    Exit
```

Operatii

Adunarea a 2 matrice A si B, rezultat in C

```
1.    If (M ≠ X) or (N ≠ Y) Then
2.          Print: Addition is not possible.
3.          Exit
      [End of If]
4.    Repeat For I = 1 to M
5.          Repeat For J = 1 to N
6.                Set C[I][J] = A[I][J] + B[I][J]
            [End of Step 5 For Loop]
      [End of Step 6 For Loop]
7.    Exit
```

Operatii

Stergere element dintr-un tablou

```
1.    Set ITEM = A[LOC]
2.    Repeat For I = LOC to N
3.          Set A[I] = A[I+1]
      [End of For Loop]
4.    Set N = N - 1
5.    Exit
```

[Setare element de sters]

[Mutare element urmator inapoi]

[Actualizare numar elemente in tablou]

Operatii

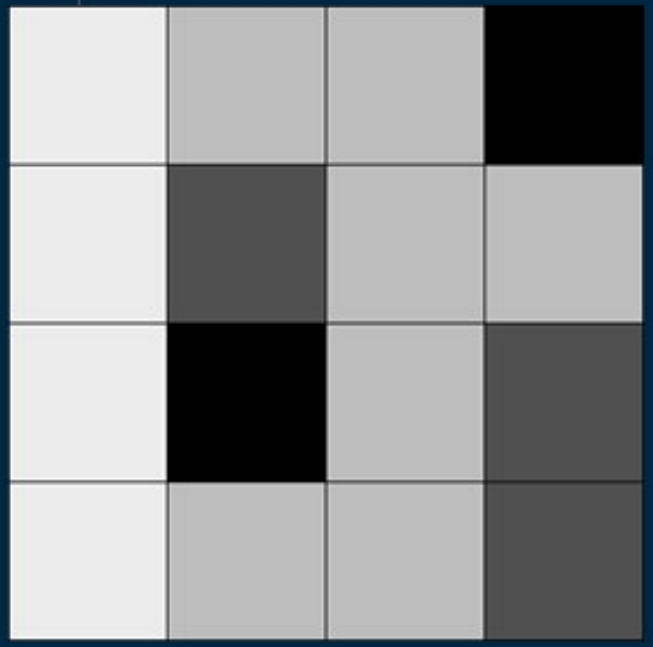
Inmultirea a 2 matrice A si B, rezultat C

```
1.    If (M ≠ Y) or (N ≠ X) Then
2.         Print: Multiplication is not possible.
3.    Else
4.         Repeat For I = 1 to N
5.              Repeat For J = 1 to X
6.                   Set C[I][J] = 0
7.                   Repeat For K = 1 to Y
8.                        Set C[I][J] = C[I][J] + A[I][K] * B[K][J]
                     [End of Step 7 For Loop]
               [End of Step 5 For Loop]
         [End of Step 4 For Loop]
     [End of If]
9.   Exit
```

Exemple

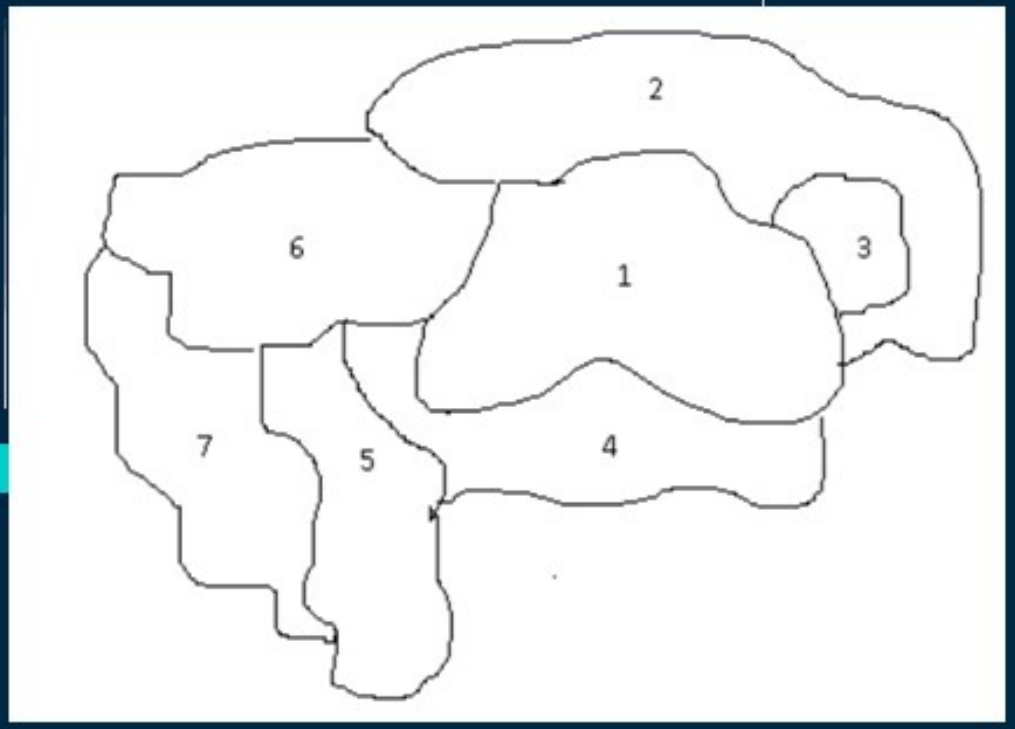
04

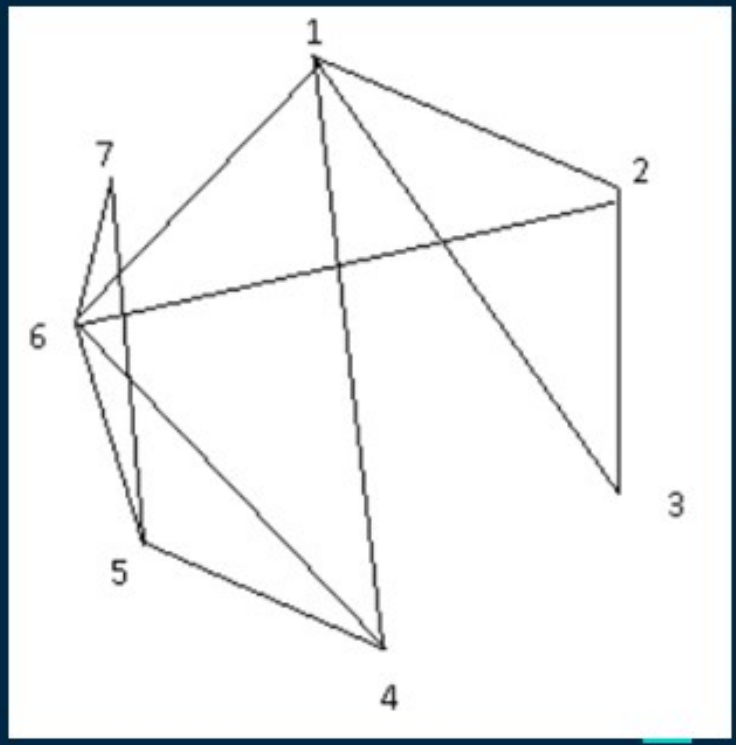
Modelare imagine



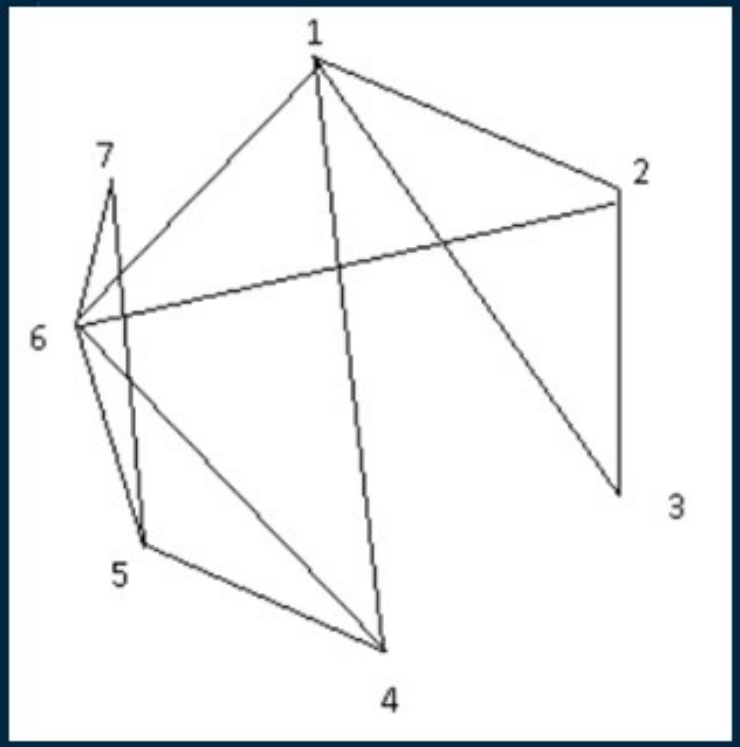
```
Integer imagine[4,4] = { {236, 189, 189, 0},
                         {236, 80, 189, 189},
                         {236, 0, 189, 80},
                         {236, 189, 189, 80} };
```

Harta bidimensionala





Harta bidimensionala




```
int harta[20][20]={
                    {0,1,1,1,0,1,0},
                    {1,0,1,0,0,1,0},
                    {1,1,0,0,0,0,0},
                    {1,0,0,0,1,1,0},
                    {0,0,0,1,0,1,1},
                    {1,1,0,1,1,0,1},
                    {0,0,0,0,1,1,0}};
```

Combinatorica

Adunare

$$A=\begin{pmatrix}2 & 3 & 0\\ 1 & 2 & -1\\ 1 & 2 & 3\end{pmatrix} \text{ si } B=\begin{pmatrix}3 & 2 & 2\\ -1 & 2 & 1\\ 1 & 2 & 2\end{pmatrix}$$

Inmultire

$$A=\begin{pmatrix}2 & 0\\ 3 & 1\\ 1 & 2\end{pmatrix}, B=\begin{pmatrix}1 & 2 & 1\\ 2 & 1 & 0\end{pmatrix}$$

$$A\cdot B=\begin{pmatrix}l1\cdot c1 & l1\cdot c2 & l1\cdot c3\\ l2\cdot c1 & l2\cdot c2 & l2\cdot c3\\ l3\cdot c1 & l3\cdot c2 & l2\cdot c3\end{pmatrix}=\begin{pmatrix}2\cdot 1+0\cdot 2 & 2\cdot 2+0\cdot 1 & 2\cdot 1+0\cdot 0\\ 3\cdot 1+1\cdot 2 & 3\cdot 2+1\cdot 1 & 3\cdot 1+1\cdot 0\\ 1\cdot 1+2\cdot 2 & 1\cdot 2+2\cdot 1 & 1\cdot 1+2\cdot 0\end{pmatrix}=\begin{pmatrix}2 & 4 & 2\\ 5 & 7 & 3\\ 5 & 4 & 1\end{pmatrix}$$

Tabla de joc

```
const XandZero2D = {
                        {‘A’, ‘N', ‘A’},
                        {‘N', ‘A’, ‘N’},
                        {‘A', ‘N', ‘A’}};
```

Joc de cărți / Board games

Gestionare pachet de carti

```
enum Value { Doi, Trei, Patru, Cinci, Sase, Sapte, Opt, Noua, Zece, Valet, Dama, Popa, As }
```

```
enum Suit { Pica, Romb, Cupa, Trefla }
```

Joc de carti

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int randInt(int n);
void generareCarti();

char *suita[4]={"trefla","pica","romb","inima"};
char *ranks[13]={"As","Doi","Trei","Patru","Cinci","Sase","Sapte","Opt","Noua",
"Zece","Valet","Dama","Popa"};

int main()
{
    int x,y;
    srand(time(NULL));
    do{
        printf("\nIntrodu un numar de carti generate ( 0 pentru Iesire)\n");
        scanf("%i",&y);
        if(y==0)
            break;
        for(x=1; x<=y; x++)
           generareCarti();
    }while(1);
    return 0;
}
```

Joc de carti

```
void generareCarti()
{
    int r,s;
    r=randInt(13);
    s=randInt(4);
    printf("%s de %s\n",ranks[r],suita[s]);
}

int randInt(int n)
{
    return rand()%n;
}
```

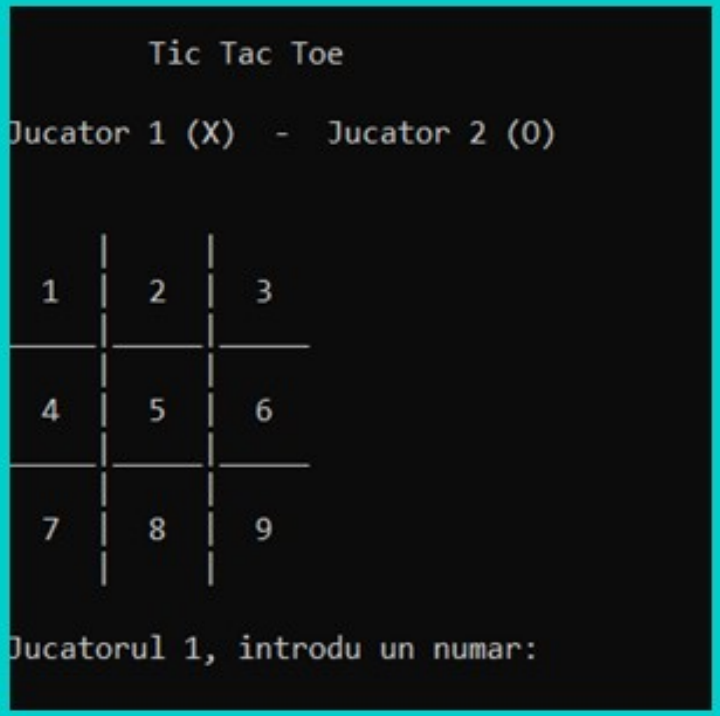
```
Introdu un numar de carti generate ( 0 pentru Iesire)
12
As de trefla
Valet de inima
As de inima
Dama de inima
As de pica
Sase de romb
Zece de inima
Trei de trefla
Zece de trefla
As de pica
Noua de trefla
Patru de romb

Introdu un numar de carti generate ( 0 pentru Iesire)
```

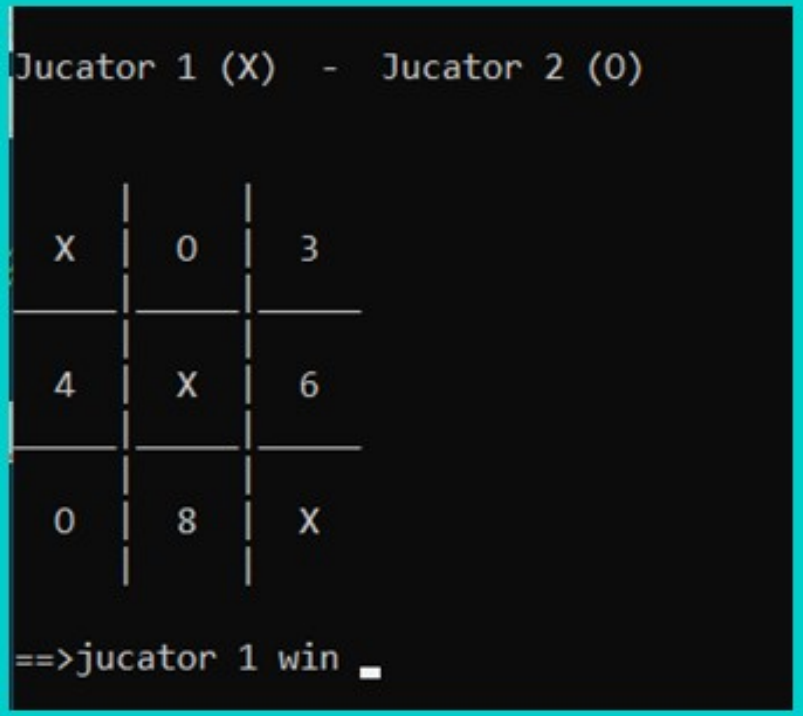
Tabla de joc

```
const XsiZero2D = {
                    {‘A’, ‘N', ‘A’},
                    {‘N', ‘A’, ‘N’},
                    {‘A', ‘N', ‘A’}};
```

Exemplu Tic Tac Toe

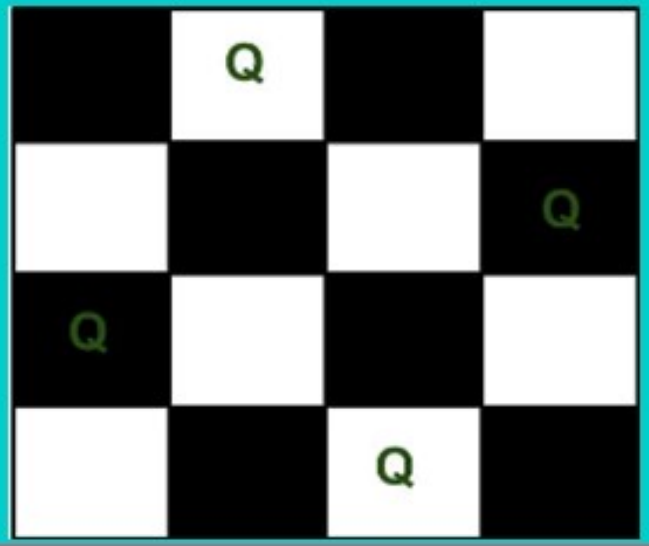
Tabla de joc


Tic Tac Toe
Jucator 1 (X) - Jucator 2 (O)
1 | 2 | 3
4 | 5 | 6
7 | 8 | 9
Jucatorul 1, introdu un numar:


Jucator 1 (X) - Jucator 2 (O)
X | 0 | 3
4 | X | 6
0 | 8 | X
==>jucator 1 win

Problema n regine

```
int board[N][N] = { { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 },
                    { 0, 0, 0, 0 } };
```



{0, 1, 0, 0}
{0, 0, 0, 1}
{1, 0, 0, 0}
{0, 0, 1, 0}

1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0

Problema “clepsidrei”

```
a b c 0 0 0
0 d 0 0 0 0
e f g 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
a b c
 d
e f g
```

Calculul sumei elementelor tabloului bidimensional care formeaza o clepsidra

```
int arr[7][7];
int sum(int stx , int sty){
    return arr[stx][sty] + arr[stx][sty+1] + arr[stx][sty+2] +
    arr[stx+1][sty+1] + arr[stx+2][sty] + arr[stx+2][sty+1] +
    arr[stx+2][sty+2];

}
int main() {
    int ans = -100;
    for(int i = 0 ; i < 6 ; i++){
        for(int j = 0 ; j < 6 ; j++){
            cin >> arr[i][j];
        }
    }

    for(int i = 0 ; i < 4 ; i++){
        for(int j = 0 ; j < 4 ; j++){
            ans = max(ans , sum(i,j));
        }
    }

    cout << ans << endl;
    return 0;
}
```

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
7
```

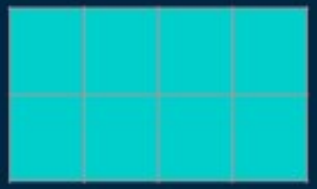
Operații aritmetice

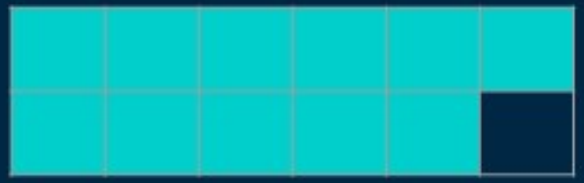
Aplicații de învățare pentru copii

Nr. linii – nr de grupuri
Nr. coloane – nr în fiecare grup

Înmulțire

3 x 4 = 12





Par / Impar

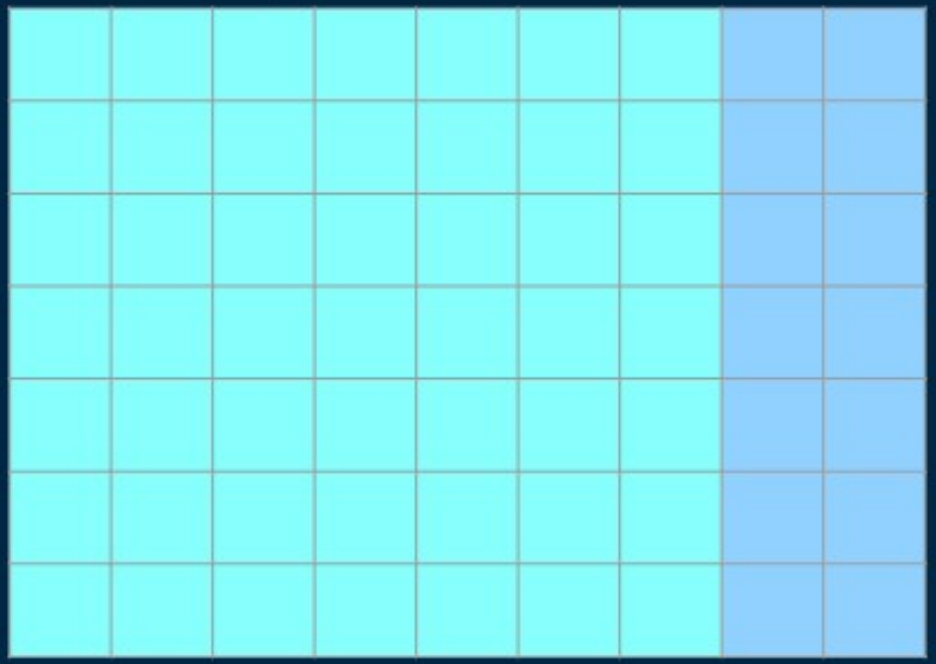
8 - par

11 - impar

Operații aritmetice

Aplicații de învățare pentru copii

$7 \times 9 = 7 \times (7+2)$
$= (7 \times 7) + (7 \times 2)$
$= 49 + 14$
$= 63$

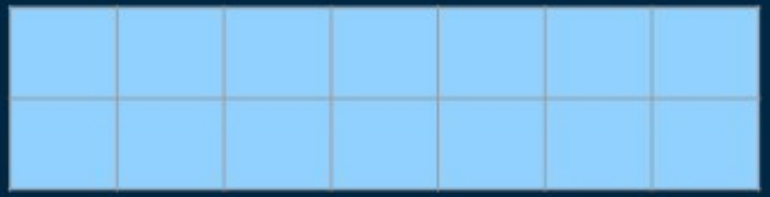


Înmulțire

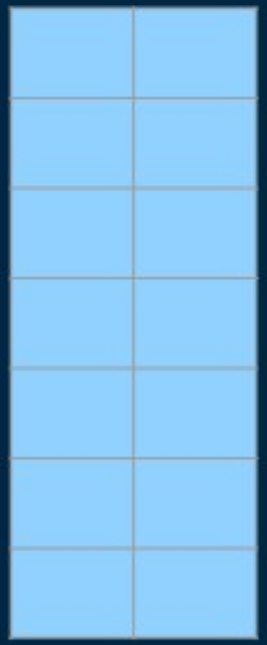
Operații aritmetice

Aplicații de învățare pentru copii

2×7 = 7 x 2



2×7

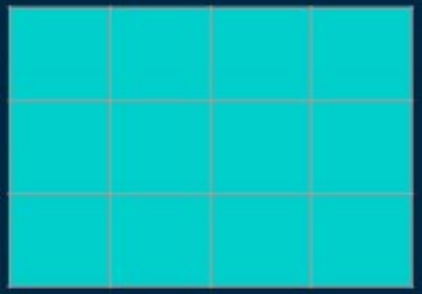


7×2

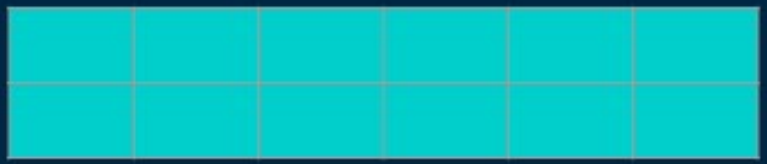
Comutativitatea

Operații aritmetice

Aplicații de învățare pentru copii



12 = 3 x 4



12 = 2 x 6

Factorizare

12 = 1 x 12

Operații aritmetice

Aplicații de învățare pentru copii

1	1
1	1

2 x 2 = 4

1	1	1
1	1	1
1	1	1

3 x 3 = 9

Numere pătrate

1	1	1
1	1	1
1	1	0

8 nu este pătrat

Tablou multidimensional

05

Declarare

Forma generală de declarare un vector multidimensional:

```
Tip nume_vector[dim1][dim2]....[dimN];
```

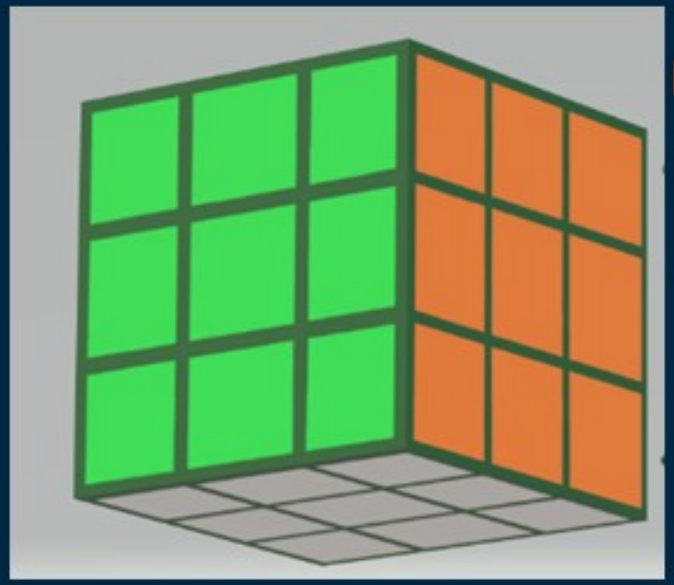
- **Tip**: tipul de date stocate în vector.
- **Nume_vector**: numele variabilei
- **dim1, dim2,... ,dimN**: mărimea fiecărei dimensiuni

Matrice: `int douaDim[10][20];`

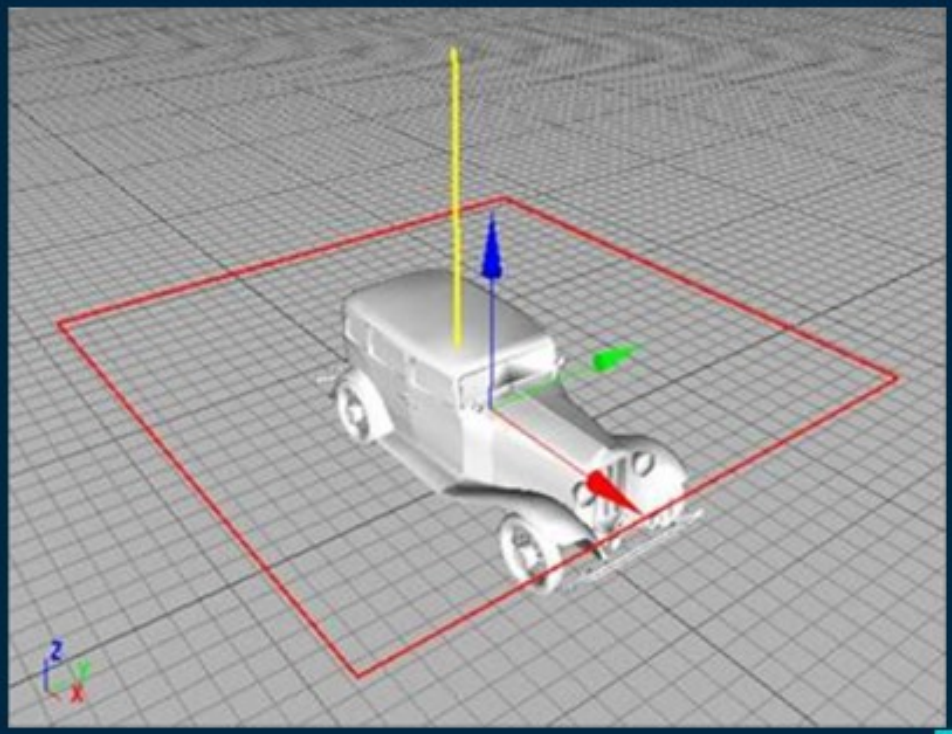
Tri-dimensional: `int treiDim[10][20][30];`

```
int test[2][3][4] = {
                        { {1, 2, 3, 4}, {4, 3, 2, 1}, {-1, -2, -3, -} },
                        { {1, 2, 3, 4}, {4, 3, 2, 1}, {-4, -3, -2, -1} }
                    };
```

Exemple

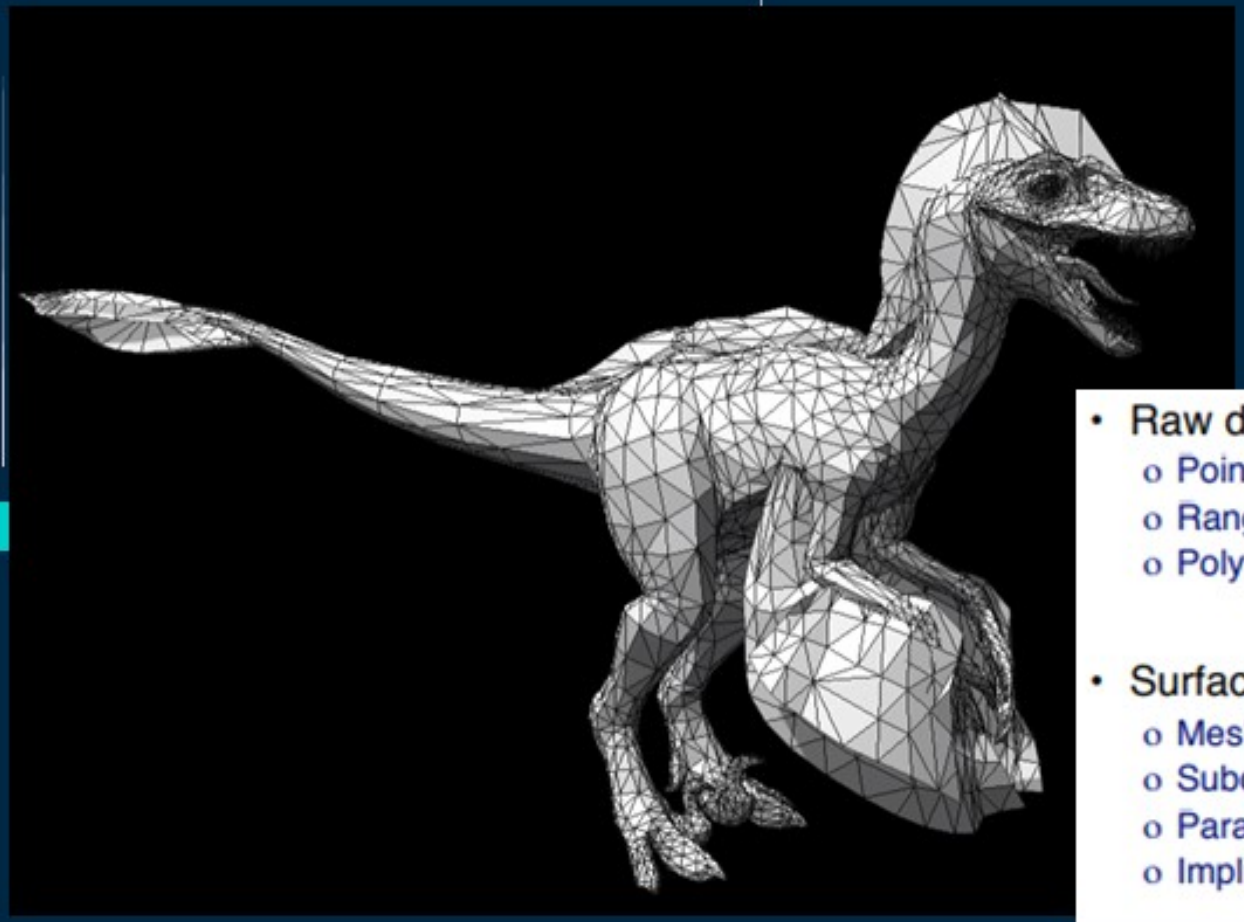


Rubik



Deplasare in spatiu

Obiecte 3D



- Raw data
  - Point cloud
  - Range image
  - Polygon soup
- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Skeleton
  - Application specific

Obiecte 3D

Vector : 1D array

Image : 2D tensor
(for each color channel)

Video : 3D tensor
(for each color channel)

https://www.antoinetlc.com/blog-summary/3d-data-representations

Intrebari?

dorin.lordache@365.univ-ovidius.ro

Mulțumesc