

Tablouri

Variabilele utilizate pana acum puteau contine o singura valoare. De multe ori e nevoie sa folosim variabile care pot stoca mai multe valori. Cu alte cuvinte se simte nevoia utilizarii tablourilor. Un tablou permite folosirea unei singure variabile pentru a stoca mai multe valori. Valorile sunt stocate la adrese consecutive cu alte cuvinte, la indecsi consecutivi incepand cu 0 . Utilizarea tablourilor ar un mare avantaj, care consta in posibilitatea utilizarii instructiunilor repetitive pentru a relucra toate valorile stocate in tablouri. Din acest motiv, tablourile mai sunt numite si "masive de date"

- **Declararea unui tablou**

Un tablou este similar cu o variabila, deci el trebuie declarat inainte de a putea fi folosit.

```
tip nume_tablou[nr_elemente]
```

Toate valorile dintr-un tablou trebuie sa fie de acelasi tip.

Sa realizam o aplicatie care cere 5 numere de la tastatura dupa care le afiseaza. Numerele vor fi memorate intr-un tablou dupa care vor fi afisate.

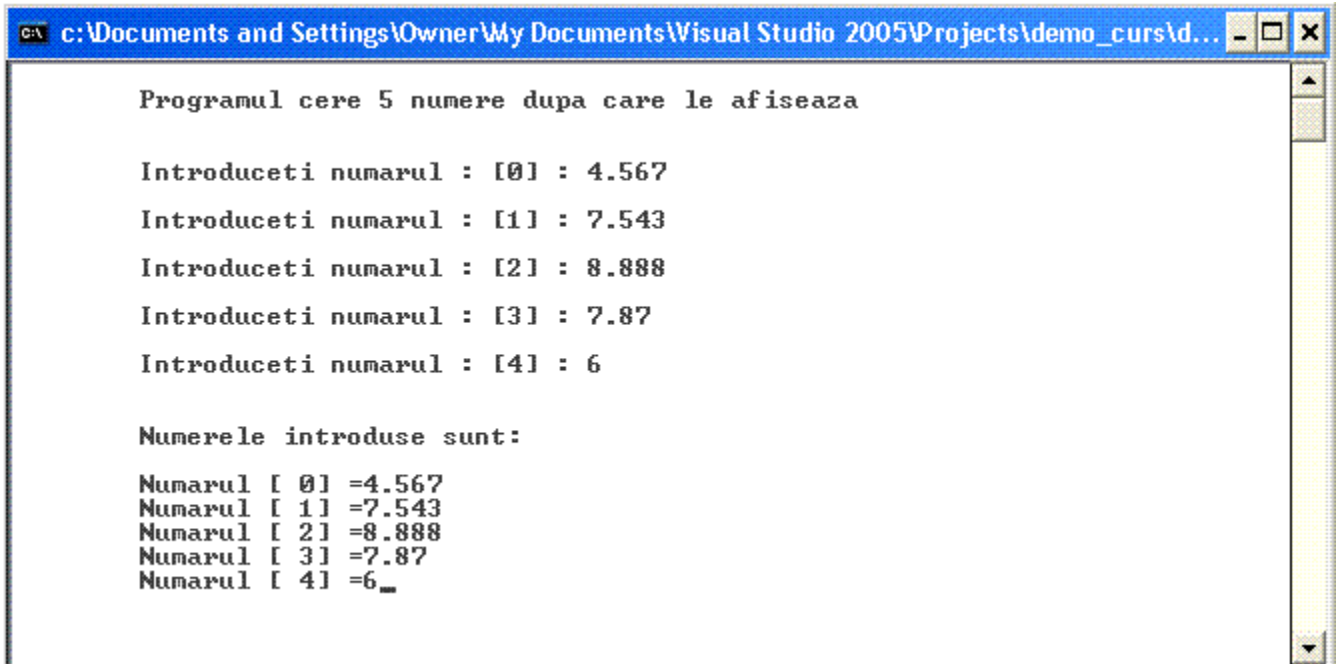
```
// Programul utilizeaza un tablou numit "numere"
// Programul cere 5 numere dupa care le afiseaza
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;

int main(void)
{
    int i=0;
    double numere[5];
    cout << "\n\tProgramul cere 5 numere dupa care le afiseaza \n\n";
    for (i=0;i<<"\n\tIntroduceti numarul : [" << i << "]" : ";
    cin >> numere[i];
    }
    cout << "\n\n\tNumerele introduse sunt: \n";
    for (i=0;i<< "\n\tNumarul [ " << i << "]" =" << numere[i] ;
    }
    cin.ignore();
    cin.get();
}
```

```
    return 0;
}
```

Daca rulam acest program, acesta ne cere cinci numere dupa care le afiseaza similar cu imaginea de jos:



```
Programul cere 5 numere dupa care le afiseaza

Introduceti numarul : [0] : 4.567
Introduceti numarul : [1] : 7.543
Introduceti numarul : [2] : 8.888
Introduceti numarul : [3] : 7.87
Introduceti numarul : [4] : 6

Numerele introduse sunt:
Numarul [ 0] =4.567
Numarul [ 1] =7.543
Numarul [ 2] =8.888
Numarul [ 3] =7.87
Numarul [ 4] =6_
```

Am definit deci o singura variabila numita "numere" insa am precizat numarul de elemente respectiv "[5]". Am creat astfel un tablou pe care putem usor sa-l manevram din instructiuni repetitive.

Putem sa cerem numarul de elemente ce vor fi introduse de la tastatura. In acest caz programul devine:

```
// Programul utilizeaza un tablou numit "numere"
// Programul cere n numere dupa care le afiseaza
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace std;

int main(void)
{
    int i,n;
    const int nmax=10;
```

```

    cout << "\n\tProgramul cere n numere dupa care le afiseaza \n\n";
    cout << " \n\tIntroduceti valoarea lui n (maxim 10):";
    cin >> n;
    double numere[nmax];
    for (i=0;i < n;i++){
        cout << "\tIntroduceti numarul : [" << i << "]" : ";
        cin >> numere[i];
    }
    cout << "\n\n\tNumerele introduse sunt:  \n";
    for (i=0;i < n;i++){
        cout << "\n\tNumarul [ " << i << "]" ="<< numere[i] ;
    }
    cin.ignore();
    cin.get();
    return 0;
}

```

• Initializarea unui tablou

Daca pentru o variabila initializarea inseamna atribuirea unei valor, pentru un tablou initializarea inseamna stabilirea dimensiunii si precizarea numarului de elemente ale tabloului. Exista doua metode de initializare a unui tablou: prin dimensionare explicita su prin dimensionare implicia.

Dimensionarea explicita a unui tablou.

Vom initializa prin dimensionare explicita, un tablou numit "luni" si vom atribui 12 valori pentru lunile din an.

Declaram tabloul de tip string, deci elementele vor fi chiar numele lunilor. Utilizand tabloul "luni", vom realiza o aplicatie in care se cere numarul lunii dupa care programul, afiseaza numele lunii.

```

// Programul utilizeaza un tablou numit "numere"
// Programul cere numarul lunii dupa care le afiseaza numele lunii
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;

int main(void)
{
    int n=1;
    string luni[12]={"ianuarie", "februarie", "martie", "aprilie", "mai",
    "iunie",

```

```

        "iulie", "august", "septembrie", "octombrie", "noiembrie",
"decembrie");
        cout << "\n\tProgramul cere luna sub forma numerica si returneaza
numele lunii \n\n";
        while (n > 0 && n <<" \n\n\tIntroduceti luna (1-12) 0= iesire:";
            cin >> n;
            if(n > 0 && n <<"\n\tLuna introdusa este : " << luni[n-1];
        }
        return 0;
}

```

Dimensionarea implicita a unui tablou.

Vom initializa prin dimensionare implicita, un tablou numit "zile_s" si vom atribui 7 valori pentru zilele din saptamana.

Declarăm tabloul de tip string, deci elementele vor fi chiar numele zilelor. Utilizand tabloul "zile_s", vom realiza o aplicatie in care se cere numarul zilei din saptamana dupa care programul, afiseaza numele zilei.

```

// Programul utilizeaza un tablou numit "zile_s"
// Programul cere numarul zilei din saptamana dupa care le afiseaza numele
zilei
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;

int main(void)
{
    int n=1;
    string zile_s[]={"luni", "marti", "miercuri", "joi", "vineri",
"sambata", "duminica"};
    cout << "\n\tProgramul cere ziua din saptamana si returneaza numele
zilei \n\n";
    while (n > 0 && n <<" \n\n\tIntroduceti numarul zilei din saptamana
(1-7) 0= iesire:";
        cin >> n;
        if(n > 0 && n <<"\n\tZiua introdusa este : " << zile_s[n-1];
    }
    return 0;
}

```

- **Atribuirea de valori elementelor tabloului**

Dupa initializarea unui tablou, se pot atribui valori fiecarui element din tablou. Citirea

elementelor tabloului se poate face numai dupa ce acestora le-au fost atribuite valori.

Sa realizam o aplicatie care initializeaza un tablou de tip double cu maxim 10 elemente, dupa care acestora li se atribuie valori aleatoare intre 0 si 100. Vom folosi pentru aceasta instructiunea **rand()** care genereaza numere intregi intre 0 si **RAND_MAX**.

Pentru a genera umere aleatoare de tip double, definim o variabila de tip **double nr** careia ii atribuim valoarea **rand()**.

In urma operatiei: **100*nr/RAND_MAX** obtinem un numar float intre 0 si 100

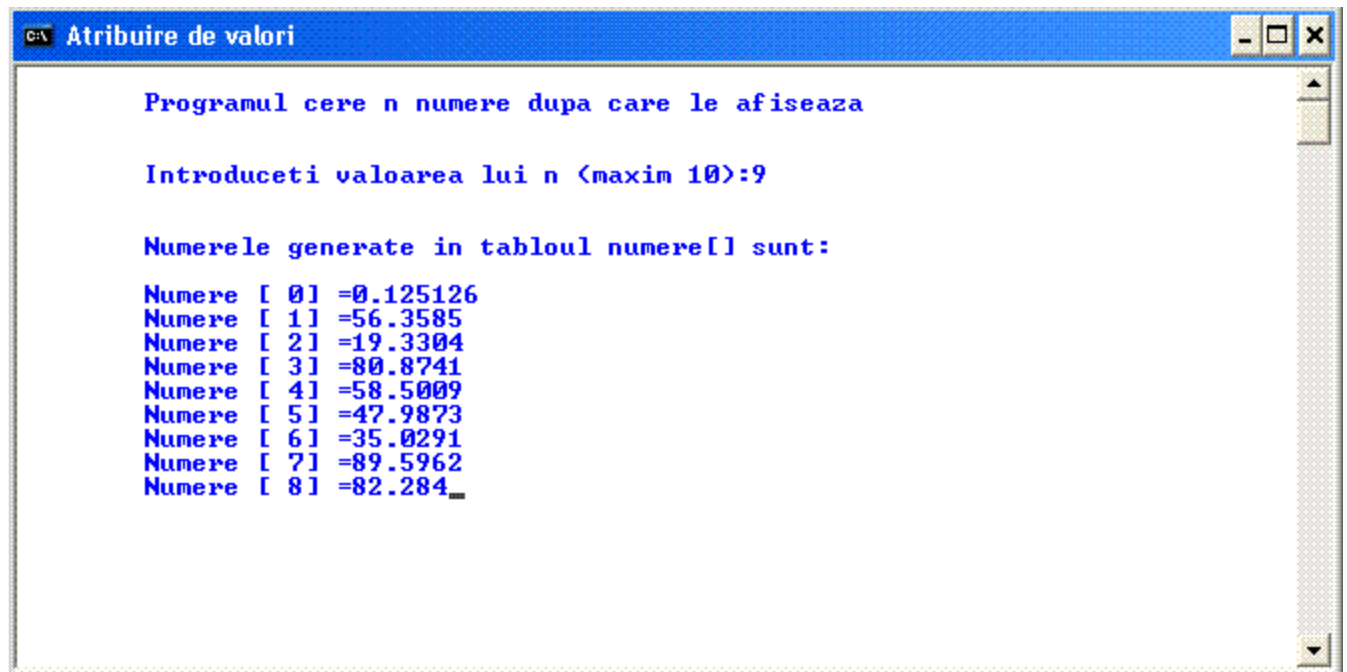
```
// Programul utilizeaza un tablou numit "numere"
// Programul genereaza n numere intre 0-100 dupa care le afiseaza
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std ;

int main(void)
{
    system("TITLE Atribuire de valori "); // Titlul ferestrei consola
    system("COLOR F9"); // Fundal alb caractere albastre
    int i,n;

    cout << "\n\tProgramul atribuie n valori aleatoare dupa care le
afiseaza \n\n";
    cout << " \n\tIntroduceti valoarea lui n (maxim 10):";
    cin >> n;
    const int nmax=10;
    double numere[nmax],nr;
    for (i=0;i < n;i++){
        nr=rand();
        numere[i]= 100*nr/RAND_MAX ; // pentru a genera numere intre 0-100
    }
    cout << "\n\n\tNumerele generate in tabloul numere[] sunt: \n";
    for (i=0;i < n;i++){
        cout << "\n\tNumere [ " << i << " ] ="<< numere[i] ;
    }
    cin.ignore();
    cin.get();
    return 0;
}
```

Dupa rularea aplicatiei si introducerea valorii 9 pentru n, obtinem:



Programul genereaza de fiecare data aceleasi numere. Pentru a genera numere diferite la fiecare rulare putem modifica programul astfel:

```
#include "stdafx.h"
#include < iostream >
#include < time.h >
using namespace std;

int main()
{
    system("TITLE Atribuire de valori aleatoare "); // Titlul ferestrei
    console
    system("COLOR F9"); // Fundal alb caractere albastre
    cout << "\n\tProgramul atribuie n valori aleatoare dupa care le
    afiseaza \n\n";
    cout << " \n\tIntroduceti valoarea lui n (maxim 10):";
    int i,n;
    const int nmax=10;
    double numere[nmax],nr;
    srand(time(NULL));
    cin >> n;
    for (i=0;i < n;i++){
        nr=rand()%100; // pentru a genera numere intre 0-100
        numere[i]= nr ;
    }
    cout << "\n\n\tNumerele generate in tabloul numere[] sunt:  \n";
    for (i=0;i < n;i++){
        cout << " \n\tNumere [ " << i << " ] ="<< numere[i] ;
```

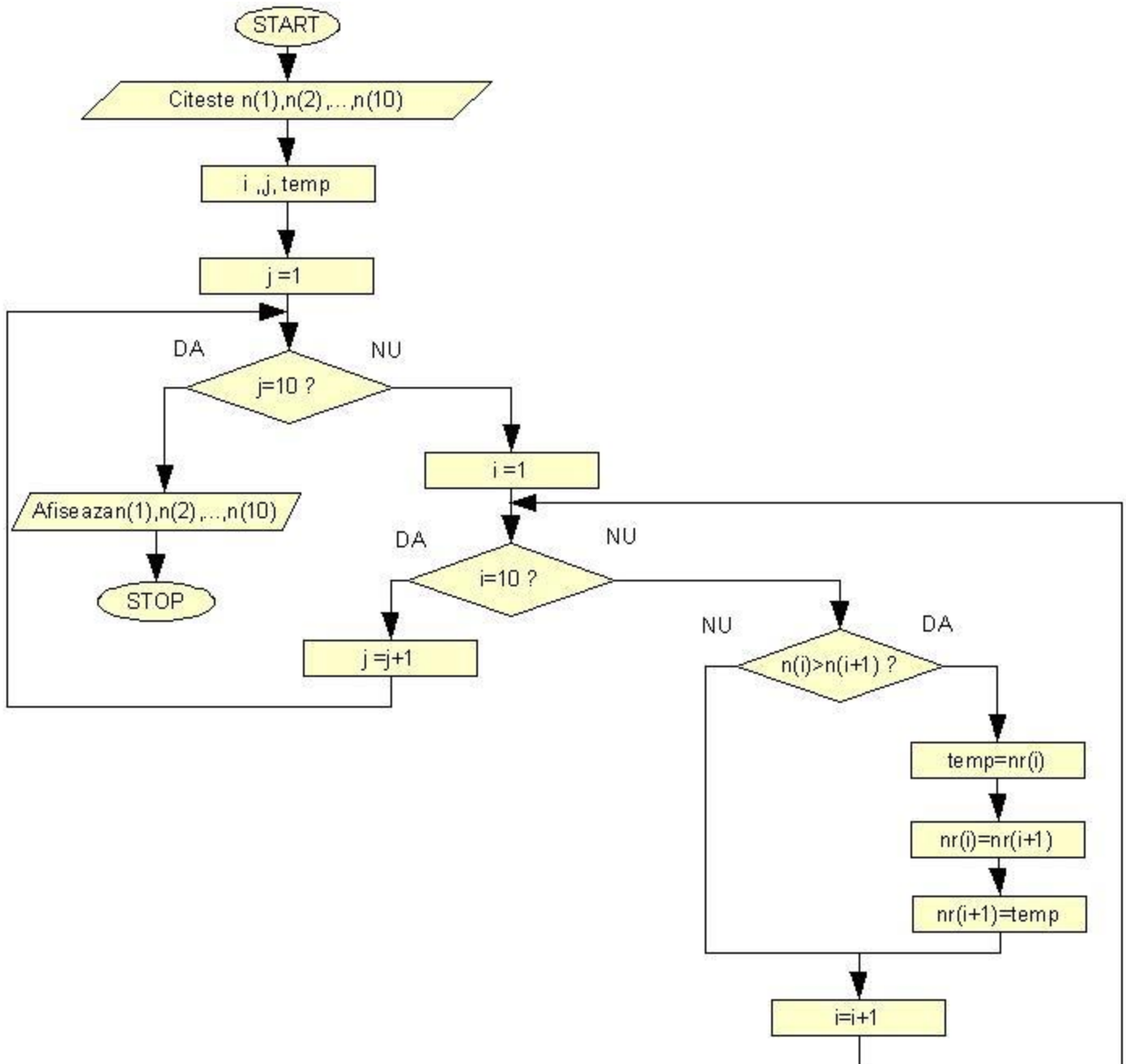
```
}  
cin.ignore();  
cin.get();  
return 0;  
}
```

Tablourile sunt des utilizate in programare. Exista o gama larga de cazuri in care sunt necesare tablourile. Sa luam de exemplu aplicatiile de ordonare siruri analizate in cadrul schemelor logice. Pentru a implementa aceste scheme avem nevoie de tablouri in care sa pastram fiecare element. Tabloul contine deci toate numerele din sir. Acest tablou este de ordinul 1 si il vom numi vector.

- **Tablouri unidimensionale (vectori)**

Vom implementa schemele logice pentru ordonarea sirurilor de numere analizate in cadrul schemelor logice.

Ordonarea sirurilor.



: Codificam schema logica de sus intr-un program C++ si obtinem:

```

// Programul utilizeaza un tablou de numere double numit "n"
// Programul cere 10 numere, le ordoneaza crescator, dupa care le afiseaza
#include "stdafx.h"
#include < iostream >
#include < string >

```



```

using namespace std;

int main(void)
{
    int i,j;
    double n[11],temp;
    cout << "\n\tProgramul cere 10 numere dupa care le ordoneaza si le
afiseaza \n\n";
    // introducere numere

    for (i=1;i<<"\tIntroduceti numarul : [" << i << "]" : ";
        cin >> n[i];
    }
    // afisarea numerelor

    cout << "\n\n\tNumerele introduse sunt: \n\n";
    for (i=1;i<< " : " << n[i] ;
    }
    // ordonarea crescatoare a numerelor
    j=1;
    while (jn[i+1]){
        temp=n[i];
        n[i]=n[i+1];
        n[i+1]=temp;
    }
    i++;
}
j++;
}
// afisarea numerelor ordonate crescator

cout << "\n\n\tNumerele ordonate sunt: \n\n";
for (i=1;i<< " : " << n[i] ;
}
cin.ignore();
cin.get();
return 0;
}

```

Pentru a respecta intru totul schema logica, nu s-a folosit elementul n[0] drept pentru care numarul de lemente al tabloului devine 11 nu 10. Aceasta metoda de ordonare se numeste metoda "bulelor"

Dupa rulara programului, obtinem:

```
c:\Documents and Settings\Owner\My Documents\Visual Studio 2005\Projects\demo_curs\d... - □ ×

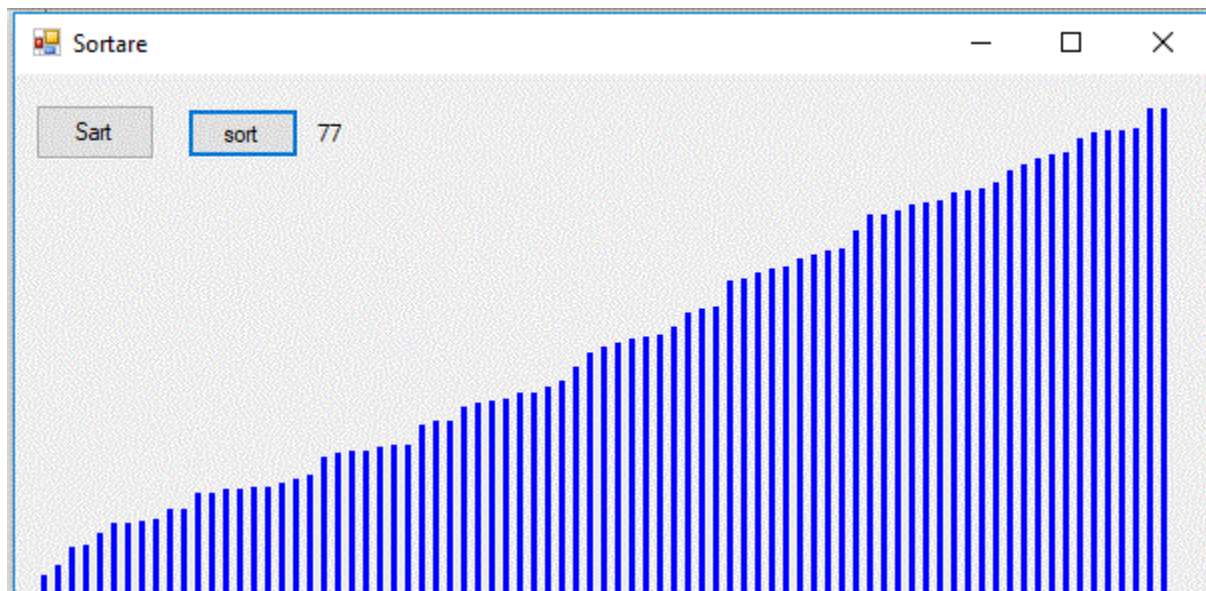
Programul cere 10 numere dupa care le ordoneaza si le afiseaza

Introduceti numarul : [1] : 77
Introduceti numarul : [2] : 88.456
Introduceti numarul : [3] : 75
Introduceti numarul : [4] : 2
Introduceti numarul : [5] : 0
Introduceti numarul : [6] : 124
Introduceti numarul : [7] : 7.89
Introduceti numarul : [8] : 3.14
Introduceti numarul : [9] : 2
Introduceti numarul : [10] : 11.11

Numerele introduse sunt:
: 77 : 88.456 : 75 : 2 : 0 : 124 : 7.89 : 3.14 : 2 : 11.11

Numerele ordonate sunt:
: 0 : 2 : 2 : 3.14 : 7.89 : 11.11 : 75 : 77 : 88.456 : 124_
```

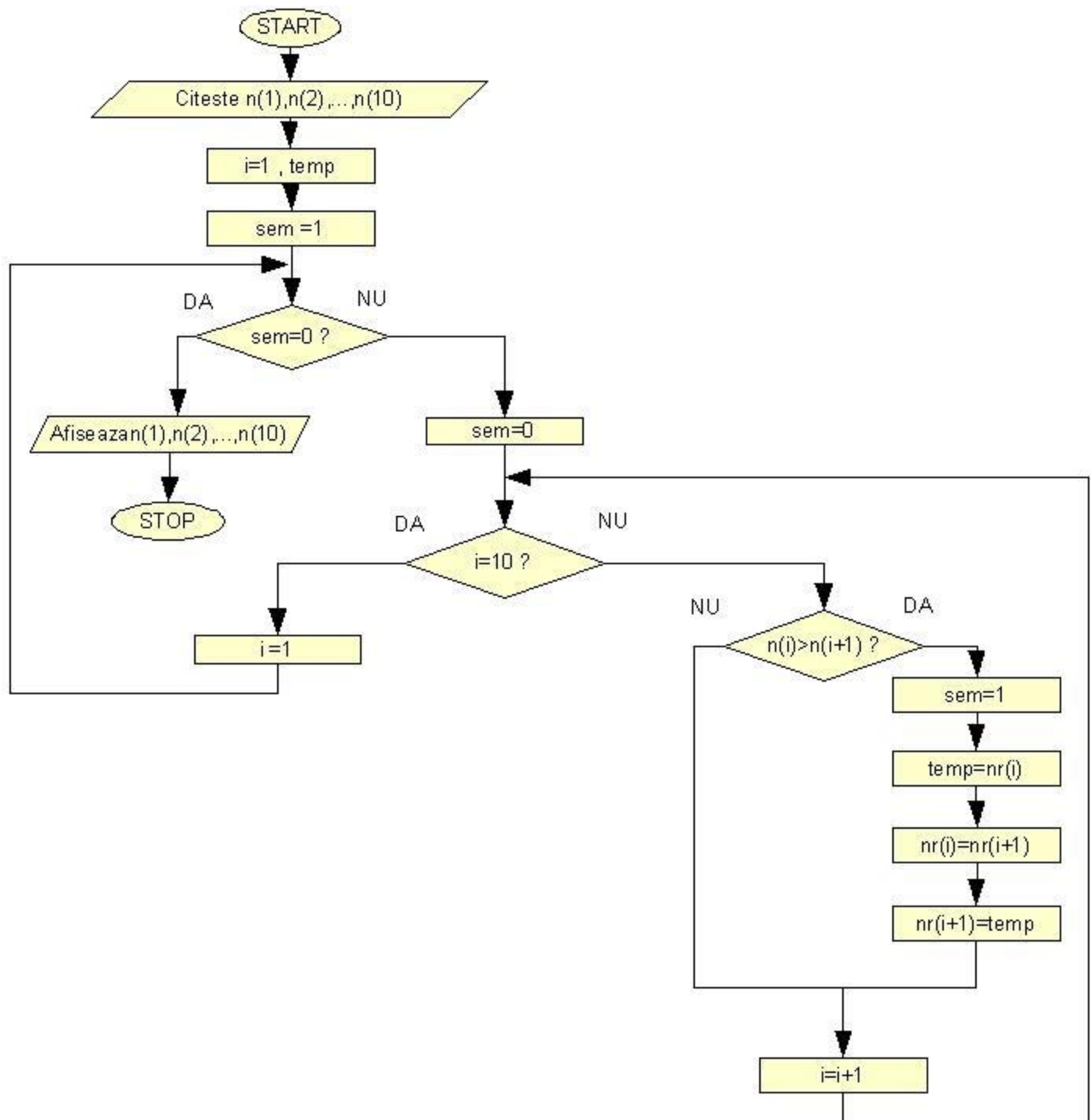
Aplicatia **sort_w** exemplifica grafic acest algoritm



Dupa rularea aplicatiei se poate observa ca algoritmul poate fi imbunatatit astfel parcurgerea unei linii nu trebuie facuta pana la capat adica pana la 11, fiind suficienta parcurgerea pana la 11-j-1 , algoritmul devenind:

```
// ordonarea crescatoare a numerelor
j=1;
while (jn[i+1]){
    temp=n[i];
    n[i]=n[i+1];
    n[i+1]=temp;
    i++;
}
j++;
}
```

Metoda poate fi perfectionata in sensul ca se introduce un semafor care va fi setat in momentul cand s-a trecut prin tot sirul si nu s-a facut nici o schimbare, deci sirul e ordonat si se poate afisa. Schema logica in acest caz devine:



```

// Programul cere 10 numere, le ordoneaza crescator dupa metoda optima, dupa
care le afiseaza
#include "stdafx.h"
#include < iostream >

```

```

#include < string >

using namespace std;

int main(void)
{
    int i,sem;
    double n[11],temp;
    cout << "\n\tProgramul cere 10 numere dupa care le ordoneaza si le
afiseaza \n\n";
    // introducere numere

    for (i=1;i<<"\tIntroduceti numarul : [" << i << "]" : ";
        cin >> n[i];
    }
    // afisarea numerelor

    cout << "\n\n\tNumerele introduse sunt: \n\n";
    for (i=1;i<< " : " << n[i] ;
    }
    // ordonarea crescatoare a numerelor
    i=1;
    sem=1;
    while (sem !=0){
        sem=0;
        while (i<10){
            temp=n[i];
            n[i]=n[i+1];
            n[i+1]=temp;
            sem=1;
        }
        i++;
    }
    i=1;
}
// afisarea numerelor ordonate crescator

    cout << "\n\n\tNumerele ordonate sunt: \n\n";
    for (i=1;i<< " : " << n[i] ;
    }
    cin.ignore();
    cin.get();
    return 0;
}

```

O alta metoda de ordonare, este metoda selectiei. Conform acestei metode, se urmeaza urmatoorii pasi:

- Se scaneaza sirul de nr elemente-1 ori (Dupa fiecare scanare se incrementeaza pozitia curenta).
- Dupa fiecare scanare se gaseste minimul si se salveaza pe pozitia curenta.

- Se introduce o var numita "sel" care la prima scanare ia valoarea primului element. In timpul scanarii, cand se gaseste un element mai mic decat "sel" se face schimbarea intre "sel" si elementul curent.
- In variabila "sel" se gaseste tot timpul minimul actual .
- Se scaneaza numai restul sirului(sirul de la pozitia curenta pana la sfarsit)
- Dupa o trecere completa prin sirul ramas, in "sel" avem minimul urmator si se poate salva in pozitia curenta

```
// ordonarea crescatoare prin metoda selectiei
// Programul cere nr_max numere, le ordoneaza crescator dupa metoda
// selectiei, dupa care le afiseaza
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;

int main(void)
{
    const int nr_max=5;
    int i,j;
    double n[nr_max+1],temp, sel;
    cout << "\n\tProgramul cere " << nr_max << " numere dupa care le
ordoneaza si le afiseaza \n\n";
    // introducere numere

    for (i=1;i < nr_max+1;i++){
        cout << "\tIntroduceti numarul : [" << i << "]" : ";
        cin >> n[i];
    }
    // afisarea numerelor

    cout << "\n\n\tNumerele introduse sunt: \n\n";
    for (i=1;i < nr_max+1;i++){
        cout << " : " << n[i] ;
    }
    // ordonarea crescatoare a numerelor
    i=1;
    j=1;
    sel=n[1];
    while (j < nr_max){
        for (i=j; i < nr_max; i++) {
            if (sel>n[i+1]){
                temp=sel;
                sel=n[i+1];
                n[i+1]=temp;
            }
        }
        n[j]=sel;
        j++;
    }
```

```

        sel=n[j];
    }
// afisarea numerelor ordonate crescator

    cout << "\n\n\tNumerele ordonate sunt: \n\n";
    for (i=1;i < nr_max+1;i++){
        cout << " : " << n[i] ;
    }
    cin.ignore();
    cin.get();
    return 0;
}

```

Utilizarea tablourilor pentru siruri de caractere vectori de caractere

Sirurile de caractere pot fi tratate ca si vectori ale carui elemente sunt caractere.

Pentru a citi de exemplu numele si prenumele cu o singura instructiune va trebui sa definim o variabila de tablou de tip caracter.

```

// Utilizarea tablourilor de caractere

#include "stdafx.h"
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;

int main(void)
{
    char nume[80]; // se initializeaza un tablou unidimensional de tip
char cu numarul de elemente 80
    cout << "\n\tIntroduceti numele si prenumele:";
    cin.getline(nume,80);
    cout << "\n\n\n\tNumele tau este: " << nume;
    cin.ignore();
    cin.get();
    return 0;
}

```

Daca nu am fi definit variabila nume de tip tablou, nu am fi putut citi numele si prenumele intr-o singura instructiune. Utilizand instructiuni de genul:

cin >> nume ; , variabila nume ar fi retinut numai caracterele introduse pana la primul caracter spatiu.

- **Tablouri bidimensionale (matrici)**

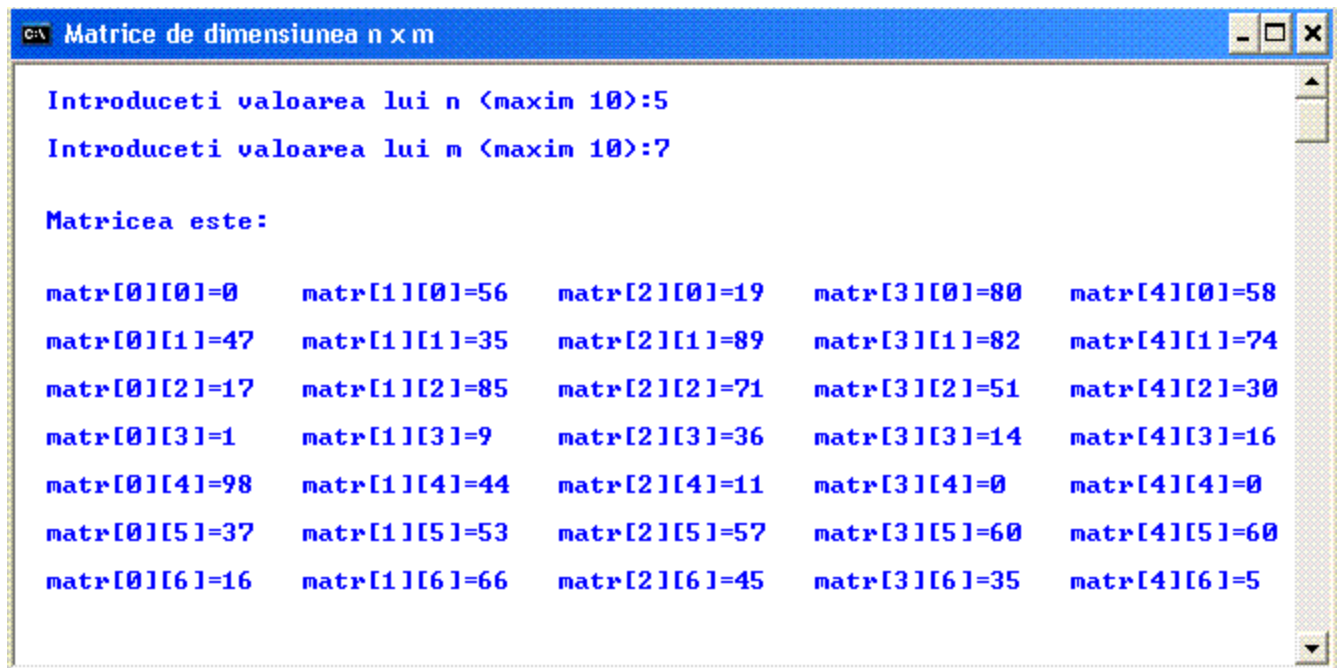
Sa initializam o matrice de tip int cu n x m elemente si sa-i atribuim valori aleatoare in domeniul 0-100;

```
// Programul utilizeaza o matrice cu n x m elemente
// Elementelor matricii li se atribuite valori intregi aleatoare intre 0 si 100
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std ;

int main(void)
{
    system("TITLE Matrice cu n x m elemente");// Titlul ferestrei
    console
    system("COLOR F9"); // Fundal alb caractere albastre
    int i,j,n,m;
    cout <<" \n\  Introduceti valoarea lui n (maxim 10):";
    cin >> n;
    cout <<" \n\  Introduceti valoarea lui m (maxim 10):";
    cin >> m;
    const int nmax=10;
    const int mmax=10;
    int matr[nmax][mmax],nr;
    for (j=0; j < m; j++) {
        for (i=0;i < n;i++){
            matr[i][j]= 100*rand()/RAND_MAX ; // pentru a genera numere
intre 0-100
        }
    }
    cout <<"\n\n\  Matricea este:  \n\n";
    for (j=0; j < m; j++) {
        cout <<"\n" ;
        for (i=0;i < n;i++){
            cout <<"  matr[" << i << "]"[" << j << "]= " <<
matr[i][j] << "\t";
        }
    }
    cin.ignore();
    cin.get();
    return 0;
}
```

Rulam aplicatia de sus si obtinem:



Sa initializam doua matrici patratice de tip int cu nXn elemente , sa atribuim apoi valori aleatoare elementelor celor doua matrici in domeniul 0-1000 dupa care sa afisam matricile si suma celor doua matrici;

```
// Programul genereaza si aduna doua matrici patratice cu nXn elemente
// Elementelor matricilor li se atribuite valori intregi aleatoare intre 0
si 1000
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std ;

int main(void)
{
    system("TITLE Adunarea a doua matrici patratice cu nxn elemente");//
    Titlul ferestrei consola
    system("COLOR F9"); // Fundal alb caractere albastre
    int i,j,n;
    cout <<" \n\n * Introduceti valoarea lui n (maxim 10):";
    cin >> n;

    const int nmax=10;
    int m1[nmax][nmax],m2[nmax][nmax];
    // generare matrice m1
    for (j=0; j < n; j++) {
        for (i=0;i < n;i++){
```

```

        m1[i][j]= 1000*rand()/RAND_MAX ; // pentru a genera numere
intre 0-1000
    }
}
// generare matrice m2
cout << "\n\ * Cele doua matrici sunt: \n\n";
for (j=0; j < n; j++) {
    for (i=0;i < n;i++){
        m2[i][j]= 1000*rand()/RAND_MAX ; // pentru a genera numere
intre 0-1000
    }
}
// afisare matricea m1 si m2
for (j=0; j < n; j++) {
    cout << "\n " ;
    for (i=0;i < n;i++){
        cout << m1[i][j] << "\t" ;
    }
    cout << "\t\t" ;
    for (i=0;i < n;i++){
        cout << m2[i][j] << "\t" ;
    }
}
// afisare m1 + m2
cout << "\n\n\ * Suma celor doua matrici este:\n\n";
for (j=0; j < n; j++) {
    cout << "\n\n\t\t\t" ;
    for (i=0;i < n;i++){
        cout << m1[i][j]+m2[i][j] << "\t" ;
    }
}

cin.ignore();
cin.get();
return 0;
}

```

Rulam aplicatia de sus si obtinem:

```
C:\ Adunarea a doua matrici de dimensiune n
* Introduceti valoarea lui n (maxim 10):4
* Cele doua matrici sunt:

1      563      193      808      91      364      147      165
585     479     350     895     988     445     119      4
822     746     174     858      8      377     531     571
710     513     303     14      601     607     166     663

* Suma celor doua matrici este:

          92      927      340      973
        1573     924      469      899
        830     1123      705     1429
        1311     1120      469      677
```

Sa initializam doua matrici de tip int cu n x m respectiv m x p elemente.

Cerem valori si le atribuim apoi elementelor celor doua matrici si le afisam.

Afisam produsul celor doua matrici.

Dupa cum se cunoaste din algebra elementara operatia de inmultire a doua matrici se realizeaza prin procedeul " linii prin coloane".

Adica elementul $c[i][j] = \text{Suma de la } k=1 \text{ la } n \text{ din } a[i][k] * b[k][j]$

```
// Programul inmulteste doua matrici a,b cu n x m respectiv m x p elemente
// Elementelor matricilor li se atribuite valori citite de la utilizator
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std ;

int main(void) {
system("TITLE Inmultirea a doua matrici a,b cu n x m respectiv m x p
elemente");// Titlul ferestrei consola
system("COLOR F9");// Fundal alb caractere albastre
int n,m,p,i,j,k,a[10][10],b[10][10],c[10][10];
cout << "\n\tIntroduceti numarul de linii n = ";
cin >> n;
cout << "\tIntroduceti numarul de coloane m = ";
cin >> m;
cout << "\n";
for(i=1;i <= n;i++)
for(j=1;j <= m;j++)
{
```

```

cout << "\ta[" << i << "," << j <<"]= ";
cin >> a[i][j];
}
cout<<"\n\tElementele matricei A sunt: "<<"\n\n";
for(i=1;i <= n;i++)
{
for(j=1;j <= m;j++)
cout << "\t" << a[i][j];
cout << "\n";
}
cout << "\n\tIntroduceti numarul de coloane p = ";
cin >> p;
cout << "\n";
for(i=1;i <= m; i++)
for(j=1;j <= p; j++)
{
cout << "\tb[" << i << "," << j <<"]= ";
cin >> b[i][j];
}
cout<<"\n\n\tElementele matricei B sunt: " << "\n\n";
for(i=1;i <= m;i++)
{
for(j=1;j <= p;j++)
cout << "\t" << b[i][j];
cout << "\n";
}
for(i=1;i <= n;i++)
for(j=1;j <= p;j++){
c[i][j]=0;
for(k=1;k <= m;k++)
c[i][j]+=a[i][k]*b[k][j];
}
cout << "\n\tElementele matricei produs " << "\n\n";
for(i=1;i <= n;i++)
{
for(j=1;j <= p;j++)
cout << "\t" << c[i][j];
cout << "\n";
}
cin.ignore();
cin.get();
return 0;
}

```

Rulam aplicatia de sus si obtinem:

```
C:\ Inmultirea a doua matrici a,b dimensiune n x m respectiv m x p

Introduceti numarul de linii n = 2
Introduceti numarul de coloane m = 3

a[1,1]= 1
a[1,2]= 2
a[1,3]= 3
a[2,1]= 4
a[2,2]= 5
a[2,3]= 6

Elementele matricei A sunt:

1      2      3
4      5      6

Introduceti numarul de coloane p = 2

b[1,1]= 1
b[1,2]= 2
b[2,1]= 3
b[2,2]= 4
b[3,1]= 5
b[3,2]= 6

Elementele matricei B sunt:

1      2
3      4
5      6

Elementele matricei produs

22      28
49      64
```

Sa incercam acelasi lucru dar de data aceasta sa generam aleator elementele matricilor

```
// Programul inmulteste doua matrici a,b cu n x m respectiv m x p elemente
// Elementelor matricilor li se atribuite aleator valori pana la 10
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std ;

int main(void){
system("TITLE Inmultirea a doua matrici a,b cu n x m respectiv m x p
elemente");// Titlul ferestrei consola
system("COLOR F9");// Fundal alb caractere albastre
int n,m,p,i,j,k,a[10][10],b[10][10],c[10][10];
cout << "\n\tIntroduceti numarul de linii n (maxim 10)= ";
cin >> n;
cout << "\tIntroduceti numarul de coloane m (maxim 10)= ";
cin >> m;
for(i=1;i <= n;i++)
```

```

for(j=1;j <= m;j++)
{
    a[i][j]=10*rand()/RAND_MAX ;
}
cout<<"\n\tElementele matricei A sunt: "<<"\n\n";
for(i=1;i <= n;i++)
{
    for(j=1;j <= m;j++)
    cout << "\t" << a[i][j];
    cout << "\n";
}
cout << "\n\tIntroduceti numarul de coloane p(maxim 10) = ";
cin >> p;
for(i=1;i <= m; i++)
for(j=1;j <= p; j++)
{
    b[i][j]=10*rand()/RAND_MAX;
}
cout<<"\n\tElementele matricei B sunt: " << "\n\n";
for(i=1;i <= m;i++)
{
    for(j=1;j <= p;j++)
    cout << "\t" << b[i][j];
    cout << "\n";
}
for(i=1;i <= n;i++)
    for(j=1;j <= p;j++){
        c[i][j]=0;
        for(k=1;k <= m;k++)
            c[i][j]+=a[i][k]*b[k][j];
    }
cout << "\n\tElementele matricei produs " << "\n\n";
for(i=1;i <= n;i++)
{
    for(j=1;j <= p;j++)
    cout << "\t" << c[i][j];
    cout << "\n";
}
cin.ignore();
cin.get();
return 0;
}

```

Rulam aplicatia de sus si obtinem:

```

C:\ Inmultirea a doua matrici a,b dimensiune n x m respectiv m x p

Introduceti numarul de linii n <maxim 10>= 3
Introduceti numarul de coloane m <maxim 10>= 5

Elementele matricei A sunt:
0      5      1      8      5
4      3      8      8      7
1      8      7      5      3

Introduceti numarul de coloane p<maxim 10> = 9

Elementele matricei B sunt:
0      0      3      1      1      9      4      1      0
0      3      5      5      6      6      1      6      4
3      0      6      7      8      5      3      8      7
9      9      5      1      4      2      8      2      7
8      9      9      6      3      2      2      8      0

Elementele matricei produs
115    132    116    70    85    61    82    94    83
152    144    178    125   139   124   121   158   124
90     96     137    113   134   108   79    139   116

```

• Tablouri ca argumente de functii

Pana acum am utilizat functii spre care se puteau transmite valori prin intermediul argumentelor. Se puteau deci trimite cate o singura valoare pentru fiecare argument. Exista posibilitatea ca tablourile sa poata fi transmise ca argumente pentru functii, deci spre functii sa se trimita o multitudine de valori.

Sa realizam o aplicatie care cere introducerea a n valori, le pastreaza intr-un tablou dupa care apeleaza o functie care inlocuieste valorile din tablou cu patratele acestora.

```
// se defineste si se apeleaza o functie care inlocuieste valorile din
tablou cu patratele acestora
// la sfarsit se afiseaza elementele sirului rezultat
#include "stdafx.h"
#include < iostream >
#include < string >

using namespace std;
void patrat(double v[],int imdex_max);
int main(void)
{
    const int nr_max=5;
    int i;
    double nr[nr_max];
    cout << "\n\tProgramul cere " << nr_max << " numere dupa care le
ridica la patrat si le afiseaza \n\n";
    // introducere numere

```

```

        for (i=0;i < nr_max;i++){
            cout <<"\tIntroduceti numarul : [" << i << "]" : ";
            cin >> nr[i];
        }
// afisarea numerelor

        cout << "\n\n\tNumerele introduse sunt: \n\n";
        for (i=0;i < nr_max;i++){
            cout << " : " << nr[i] ;
        }
// calcularea patratelor numerelor
        patrat(nr,nr_max);

// afisarea patratelor numerelor

        cout << "\n\n\tPatratele numerelor sunt: \n\n";
        for (i=0;i < nr_max;i++){
            cout << " : " << nr[i] ;
        }
        cin.ignore();
        cin.get();
        return 0;
}

void patrat(double v[], int index_max)
{
    for (int i=0; i < index_max; i++){
        v[i]=v[i]*v[i];
    }
}

```

Sa reluam aplicatiile pentru ordonarea sirurilor dar de data aceasta sa folosim o functie care cauta pozitia minimului intr-un sir.

Vom cauta minimul in sirul initial si il plasam pe prima pozitie, iar valoarea de pe prima pozitie o salvam in pozitia minimului gasit

Ne positionam pe urmatoarea pozitie in sir si repetam operatia de cautare a minimului. Cautarea o facem din pozitia curenta nu pe tot sirul.

Continuam operatiile de sus pana pozitia curenta este la capatul sirului.

Metoda se numeste InsertSort si programul de jos o foloseste pentru ordonarea crescatoare a unui sir.

```

// ordonarea crescatoare prin metoda InsertSort
// se defineste si se apeleaza o functie cu argument tablou
// programul cere nr_max numere, le ordoneaza crescator dupa metoda
InsertSort
// la sfarsit se afiseaza sirul ordonat crescator
#include "stdafx.h"
#include < iostream >

```



```

#include < string >

using namespace std;
int caut_min(double v[], int index_i, int index_max);
int main(void)
{
    const int nr_max=5;
    int i,j;
    double nr[nr_max+1],temp, sel;
    cout << "\n\tProgramul cere " << nr_max << " numere dupa care le
ordoneaza si le afiseaza \n\n";
    // introducere numere

    for (i=1;i < nr_max+1;i++){
        cout << "\tIntroduceti numarul : [" << i << "]" : ";
        cin >> nr[i];
    }
    // afisarea numerelor

    cout << "\n\n\tNumerele introduse sunt: \n\n";
    for (i=1;i < nr_max+1;i++){
        cout << " : " << nr[i] ;
    }

    // ordonarea crescatoare a numerelor metoda InsertSort

    for (i=1; i < nr_max+1; i++){
        int poz=caut_min(nr, i, nr_max+1);
        double aux=nr[i];
        nr[i]=nr[poz];
        nr[poz]=aux;
    }
    // afisarea numerelor ordonate crescator

    cout << "\n\n\tNumerele ordonate sunt: \n\n";
    for (i=1;i < nr_max+1;i++){
        cout << " : " << nr[i] ;
    }
    cin.ignore();
    cin.get();
    return 0;
}

int caut_min(double v[], int index_i, int index_max)
// cauta elementul minim, incepnd de la pozitia indexIni, inclusiv
// functia intoarce pozitia minimului gasit
{
    double min=v[index_i];
    int poz_m=index_i;
    for (int i=index_i; i < index_max; i++){
        if (v[i] <= min){
            min=v[i];
            poz_m=i;
        }
    }
    return poz_m;
}

```

```
}
```

- **Tablouri in spatiul de nume System**

In spatiul de nume System utilizarea tablourilor este ceva mai complicata, un tablou se declarandu-se astfel:

```
array < tipul > nume_tablou;
```

Se creaza apoi o instanta a obiectului tablou definit sus, instanta de tipul gcnew.

```
nume_tablou = gcnew array < tipul > (nr_elemente);
```

Cele doua declaratii pot fi unite intr-o singura instructiune

```
array < tipul > nume_tablou = gcnew array < tipul > (nr_elemente);
```

Vom folosi acum tablouri in spatiul System. Urmatoarea aplicatie initializeaza un tablou cu 4 numere intregi dupa care le afiseaza

```
// Programul initializeaza o matrice si atribuie patru valori pentru primele  
4 elemente  
// La sfarsit se afiseaza cele patru valori  
#include "stdafx.h"  
  
using namespace System;  
  
int main(void)  
{  
    int i;  
    array < int >^ nr = gcnew array < int > (4) {9,10,11,12};  
    Console::WriteLine("Numerele sunt: ");  
    for (i=0;i
```

Tot in spatiul System scriem o noua aplicatie care cere data si ora dupa care o afiseaza, folosind numele lunii nu numarul acesteia.

```

// Programul utilizeaza un tablou numit "luni"
// Programul cere data sistemului, apoi afiseaza data incluzand numele lunii
#include "stdafx.h"

using namespace System;

int main(void)
{
    array < String^ >^ luni = gcnew array < String^ > (12) {"ianuarie",
"februarie", "martie",
    "aprilie", "mai", "iunie", "iulie", "august", "septembrie",
"octombrie",
    "noiembrie", "decembrie"};
    System::DateTime data=System::DateTime::Now;
    int zi=System::DateTime::Now.Day;
    int luna=System::DateTime::Now.Month;
    int an=System::DateTime::Now.Year;
    Console::WriteLine("Astazi e: "+data);
    Console::WriteLine("Ziua: "+zi);
    Console::WriteLine("Luna: "+luni[luna-1]);
    Console::WriteLine("Anul: "+an);
    Console::ReadLine();
    return 0;
}

```

Totusi pentru tablouri de tip standard adica tipuri care nu sunt gestionate special de spatiul de nume System se pot folosi in continuare tablouri definite pentru spatiul std. Pentru exemplificare vom scrie o aplicatie in spatiul de nume System care utilizeaza tablouri de tip double de exemplu.

```

// Programul utilizeaza un tablou numit "nr" cu 10 elemente
// Se atribuie elementelor acestui tablou diferite valori
// Se afiseaza elementele tabloului
#include "stdafx.h"

using namespace System;

int main(void)
{
    int i;
    double nr[10];
    for (i=0;i

```

Sa simulam acum consumurile zilnice de energie in cadrul unei institutii. Consumurile pe cele 31 de zile vor fi generate aleator Valorile generate vor fi pastrate intr-un vector "consum" cu 31

elemente. Vom reprezenta grafic aceste consumuri, dupa care vom calcula valoarea medie a consumului utilizand valorile pastrate in vectorul "consum"

Deschidem un nou proiect Windows Forms Application intitulat "consum_en" pe care plasam un obiect de tip button numit button1. Completam procedura deschisa pe evenimentul Click al obiectului button1 cu :

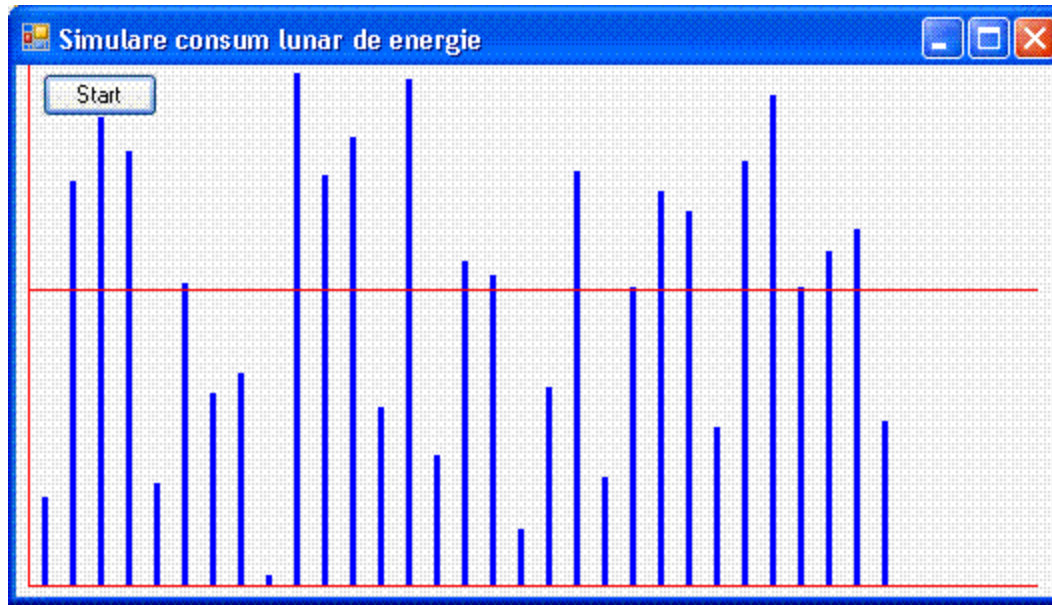
C++

```
int i=0;
const int nr_z=31;
int poz_x=14; // pozitia curenta pe axa x
float w_r=1,w_a=3; // grosimea(in pixeli) a liniei rosii respectiv
albastru
double consum[nr_z]; // vector ce pastreaza consumurile pe nr_z zile
double v_rand; // valoare aleatoare
double c_lun; // consum lunar
double c_med; // consum lunar mediu
System::Drawing::Graphics^ Desen;
Desen = this->CreateGraphics();
System::Drawing::Pen^ Creion_rosu;
Creion_rosu=gcnew
System::Drawing::Pen(System::Drawing::Color::Red,w_r);
System::Drawing::Pen^ Creion_albastru;
Creion_albastru=gcnew
System::Drawing::Pen(System::Drawing::Color::Blue,w_a);
System::Random^ n = gcnew System::Random();
Desen->Clear(System::Drawing::Color(this->BackColor));
Desen->DrawLine( Creion_rosu,6,0,6,this->Height-40);
Desen->DrawLine( Creion_rosu,6,this->Height-40,this->Width-20,this-
>Height-40);
for ( int i=0; i < nr_z; i++){
    v_rand=n->Next(this->Height-40); // se genereaza o valoare
aleatoare
    Desen->DrawLine( Creion_albastru,poz_x,this->Height-
40,poz_x,Height-40-v_rand);
    poz_x+=14;
    consum[i]=v_rand;
}
c_lun=0;
for ( int i=0; i < nr_z; i++){
    c_lun=c_lun+consum[i];
}
//calculez si afisez consumul mediu
c_med=c_lun/nr_z;
Desen->DrawLine( Creion_rosu,6,this->Height-40-c_med,this->Width-
20,this->Height-40-c_med);
delete Creion_rosu;
delete Creion_albastru;
delete Desen;
delete n;
```

C#

```
int i=0;
const int nr_z=31;
int poz_x=14; // pozitia curenta pe axa x
float w_r=1,w_a=3; // grosimea(in pixeli) a liniei rosii
respectiv albastre
float[] consum= new float[nr_z]; // vector ce pastreaza
consumurile pe nr_z zile
float v_rand; // valoare aleatoare
float c_lun; // consum lunar
float c_med; // consum lunar mediu
System.Drawing.Graphics Desen;
Desen = this.CreateGraphics();
System.Drawing.Pen Creion_rosu;
Creion_rosu=new
System.Drawing.Pen(System.Drawing.Color.Red,w_r);
System.Drawing.Pen Creion_albastru;
Creion_albastru=new
System.Drawing.Pen(System.Drawing.Color.Blue,w_a);
System.Random n = new System.Random();
Desen.Clear(this.BackColor);
Desen.DrawLine( Creion_rosu,6,0,6,this.Height-40);
Desen.DrawLine( Creion_rosu,6,this.Height-40,this.Width-
20,this.Height-40);
for ( i=0; i < nr_z; i++){
    v_rand=n.Next(this.Height-40); // se genereaza o
valoare aleatoare
    Desen.DrawLine( Creion_albastru,poz_x,this.Height-
40,poz_x,Height-40-v_rand);
    poz_x+=14;
    consum[i]=v_rand;
}
c_lun=0;
for ( i=0; i < nr_z; i++){
    c_lun=c_lun+consum[i];
}
//calculez si afisez consumul mediu
c_med=c_lun/nr_z;
Desen.DrawLine( Creion_rosu,6,this.Height-40-
c_med,this.Width-20,this.Height-40-c_med);
```

In imaginea de jos, consumul mediu a fost trasat cu linie rosie.



În aplicația de sus s-a utilizat un tablou definit în mod clasic. Din acest motiv, nu se pot apela metodele sau proprietățile instanțelor `Array`.

Utilizând tablouri definite în spațiul `System` putem beneficia de metodele și proprietățile instanțelor `Array`. Putem de exemplu să redimensionăm un tablou după ce el a fost inițializat. Astfel pentru cazul de probleme în care se cere dimensiunea tabloului, acesta poate fi redimensionat la valoarea cerută și evităm situațiile din exemplele anterioare când se inițializa un tablou și se dădea dimensiunea maximă.

Să realizăm o simplă aplicație care cere numărul de elemente al tabloului după care atribuie valori elementelor tabloului.

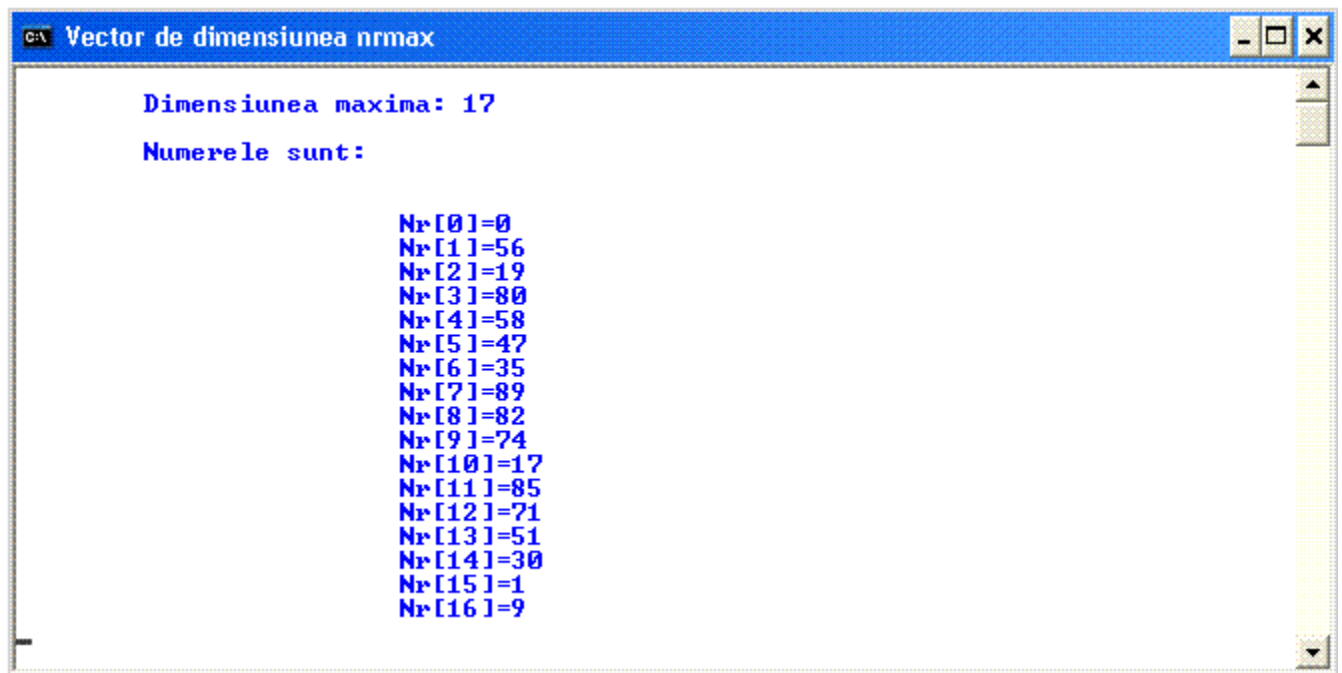
```
// Programul initializeaza un vector cu 0 elemente
// Cere numarul de elemente
// Redimensioneaza vectorul
// Atribuire valori aleatoare intre 0-100 elementelor vectorului
// La sfarsit se afiseaza elementele vectorului
#include "stdafx.h"
#include <iostream>
using namespace std;
using namespace System;

int main(void)
{
    system("TITLE Vector cu nrmax elemente");// Titlul ferestrei consola
    system("COLOR F9");// Fundal alb caractere albastre
    int i, nrmax ;
    array<int>^ numere = gcnew array<int>(0);
    Console::Write("\n\t numarul maxim de elemente: ");
    nrmax=System::Convert::ToInt16(Console::ReadLine()); // Citirea noii
    dimensiuni
```

```

        Array::Resize(numere,nrmax);                //
Redimensionare tablou
        for (i=0;i < nrmax; i++)                    // Atribuire
valori aleatoare intre 0-100
            numere[i]=100*rand()/RAND_MAX;
        Console::WriteLine("\n\tNumerele sunt: \n\n");    //
Afisare numere
        for (i=0;i < nrmax; i++)
            Console::WriteLine("\t\tNr["+i+"]="+numere[i]);
        Console::ReadLine();
        return 0;
}

```



```

C:\ Vector de dimensiunea nrmax

Dimensiunea maxima: 17

Numerele sunt:

Nr[0]=0
Nr[1]=56
Nr[2]=19
Nr[3]=80
Nr[4]=58
Nr[5]=47
Nr[6]=35
Nr[7]=89
Nr[8]=82
Nr[9]=74
Nr[10]=17
Nr[11]=85
Nr[12]=71
Nr[13]=51
Nr[14]=30
Nr[15]=1
Nr[16]=9

```

Pentru atribuirea de valori elementelor vectorului s-a utilizat rand() din spatiul de nume std. Se poate utiliza generatorul de numere aleatoare din spatiul System, caz in care aplicatia devine:

```

// Programul initializeaza un vector cu 0 elemente
// Cere numarul de elemente
// Redimensioneaza vectorul
// Atribuire valori aleatoare intre 0-100 elementelor vectorului
// La sfarsit se afiseaza elementele vectorului
#include "stdafx.h"
#include < iostream >
using namespace std;
using namespace System;

int main(void)

```

```

{
    system("TITLE Vector cu nrmax de elemente "); // Titlul ferestrei
consola
    system("COLOR F9"); // Fundal alb caractere albastre
    int i, nrmax ;
    array < int >^ numere = gcnew array < int > (0);
    System::Random^ n = gcnew System::Random();
    Console::Write("\n\tNumarul maxim de elemente: ");
    nrmax=System::Convert::ToInt16(Console::ReadLine()); // Citirea noii
dimensiuni
    Array::Resize(numere,nrmax); //
Redimensionare tablou
    for (i=0;i < nrmax; i++) // Atribuire
valori aleatoare intre 0-100
        numere[i]=n->Next(100);
    Console::WriteLine("\n\tNumerele sunt: \n\n"); //
Afisare numere
    for (i=0;i < nrmax; i++)
        Console::WriteLine("\t\t\tNr["+i+"]="+numere[i]);
    Console::ReadLine();
    return 0;
}

```

Sa reluam aplicatia pentru trasarea consumului mediu dar sa afisam consumurile zilnice pe evenimentul Klik al unui buton, iar afisarea consumului mediu sa se faca pe evenimentul Click al altui buton. Va trebui sa definim vectorul consum[nr_z], inaintea procedurilor lansate pe diverse evenimente, deoarece vectorul consum[nr_z] trebuie sa fie vizibil in toate procedurile deschise pe evenimente. La fel se intampla cu instantele desen, creion_albastru, creion_rosu, etc. Pentru a declara vectorul consum trebuie sa-l declaram de tip System in caz contrar nu putem sa-l declaram si sa fie vizibil pentru evenimentele scrise pe proceduri.

Deschidem un nou proiect Windows Forms Application intitulat "consum_med" si plasam doua obiecte de tip button numite button1 respectiv button2. Schimbam atributul text al obiectului button1 cu "Consumuri" iar atributul text al obiectului button2 cu "Consum mediu"

Consumul mediu nu se poate calcula pana nu se lanseaza "Consumuri", din aceasta cauza butonul "Consum mediu" va fi setat cu atributul "Enabled"=false. El va fi activat numai dupa lansarea procedurii de pe evenimentul Klik al butonului "Consumuri". Completam #pragma region cu :

C++

```

static int i=0;
static const int nr_z=31;
static float w_r=1,w_a=3; // grosimea(in pixeli) a liniei rosii respectiv
albastre
static array < double >^ consum = gcnew array < double > (0);
static System::Drawing::Graphics^ Desen;
static System::Drawing::Pen^ Creion_albastru ;
static System::Drawing::Pen^ Creion_rosu ;

```



```
double med(array < double >^ cons,int nrm){
    double sum=cons[0];
    for (int i=0;i < nrm; i++){
        sum=sum+cons[i];
    }

    return sum;
}
```

C#

```
static int i=0;
static const int nr_z=31;
static float w_r=1,w_a=3; // grosimea(in pixeli) a liniei rosii respectiv
albastre
static array < double > consum = new array < double > (0);
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_albastru ;
static System.Drawing.Pen Creion_rosu ;

double med(array < double > cons,int nrm){
    double sum=cons[0];
    for (int i=0;i < nrm; i++){
        sum=sum+cons[i];
    }
    return sum;
}
```

Completam procedura deschisa pe evenimentul Klik al obiectului button1 cu :

C++

```
double v_rand; // valoare aleatoare
int poz_x=10;
Array::Resize(consum,nr_z); // redimensionare la nr_z elemente
Desen = this->CreateGraphics();
Creion_rosu=gcnew
System::Drawing::Pen(System::Drawing::Color::Red,w_r);
Creion_albastru=gcnew
System::Drawing::Pen(System::Drawing::Color::Blue,w_a);
System::Random^ n = gcnew System::Random();
Desen->Clear(System::Drawing::Color(this->BackColor));
Desen->DrawLine( Creion_rosu,6,0,6,this->Height-40);
```

```

        Desen->DrawLine( Creion_rosu,6,this->Height-40,this->Width-20,this-
>Height-40);
        for ( int i=0; i < nr_z; i++){
            v_rand=n->Next(this->Height-40); // se genereaza o valoare
aleatoare
            Desen->DrawLine( Creion_albastru,poz_x,this->Height-
40,poz_x,Height-40-v_rand);
            poz_x+=14;
            consum[i]=v_rand;
        }
        this->button2->Enabled=true;

```

C#

```

        double v_rand; // valoare aleatoare
        int poz_x=10;
        Array.Resize(consum,nr_z); // redimensionare la nr_z elemente
        Desen = this.CreateGraphics();
        Creion_rosu=new System.Drawing.Pen(System.Drawing.Color.Red,w_r);
        Creion_albastru=new
System.Drawing.Pen(System.Drawing.Color.Blue,w_a);
        System.Random n = new System.Random();
        Desen.Clear(this.BackColor);
        Desen.DrawLine( Creion_rosu,6,0,6,this.Height-40);
        Desen.DrawLine( Creion_rosu,6,this.Height-40,this.Width-
20,this.Height-40);
        for ( int i=0; i < nr_z; i++){
            v_rand=n.Next(this.Height-40); // se genereaza o valoare
aleatoare
            Desen.DrawLine( Creion_albastru,poz_x,this.Height-
40,poz_x,Height-40-v_rand);
            poz_x+=14;
            consum[i]=v_rand;
        }
        this.button2.Enabled=true;

```

Completam procedura deschisa pe evenimentul Klik al obiectului button2 cu :

C++

```

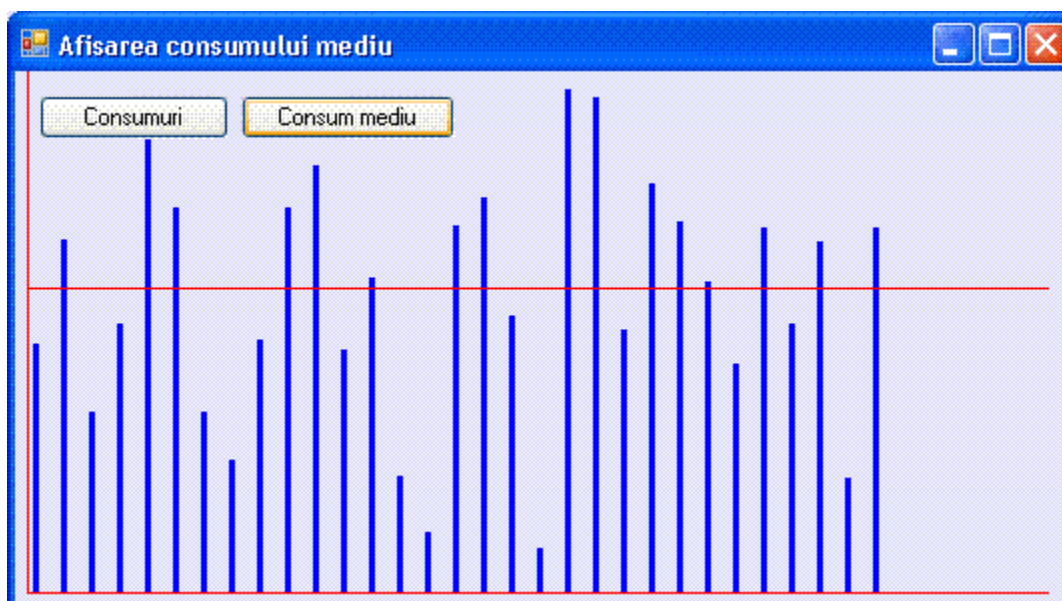
        double c_med; // consum lunar mediu
        c_med=med(consum,nr_z)/nr_z;
        Desen->DrawLine( Creion_rosu,6,this->Height-40-c_med,this->Width-
20,this->Height-40-c_med);

```

C#

```
double c_med; // consum lunar mediu  
c_med=med(consum,nr_z)/nr_z;  
Desen.DrawLine( Creion_rosu,6,this.Height-40-c_med,this.Width-  
20,this.Height-40-c_med);
```

În imaginea de jos, consumul mediu va fi trasat cu linie roșie numai pe apăsarea butonului "Consum mediu".



- **Tablouri bidimensionale în spațiul de nume System - Matrici**

Să realizăm o aplicație în spațiul System care inițializează și afișează un tablou bidimensional.

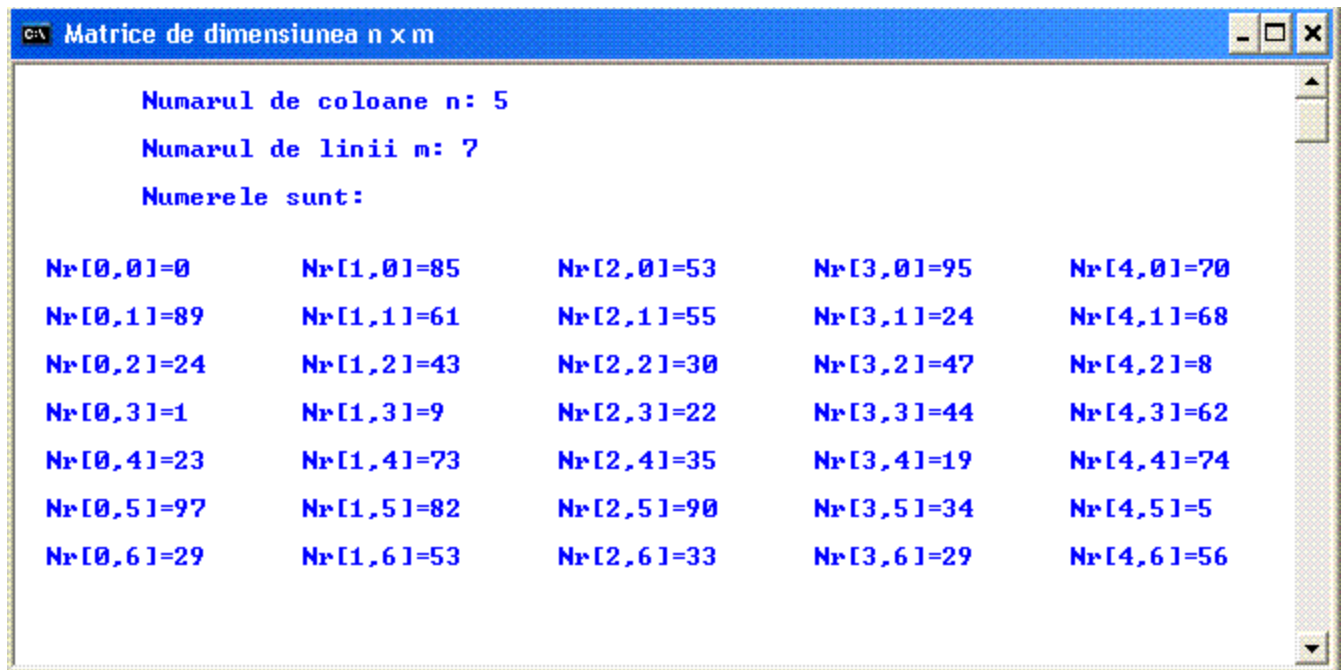
```
// Programul inițializează o matrice cu n x m elemente  
// Cere numărul de linii și coloane  
// Atribuire valori aleatoare între 0-100 elementelor matricii  
// La sfârșit se afișează elementele matricii  
#include "stdafx.h"
```

```

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

int main(void)
{
    system("TITLE Matrice cu n x m  elemente");// Titlul ferestrei
consola
    system("COLOR F9"); // Fundal alb caractere albastre
    int i,j, n, m ;
    array < int,2 >^ numere = gcnew array < int,2 > (10,10);
    System::Random^ nr = gcnew System::Random();
    Console::Write("\n\tNumarul de coloane n: ");
    n=System::Convert::ToInt16(Console::ReadLine());    // Citirea noii
dimensiuni n
    Console::Write("\n\tNumarul de linii m: ");
    m=System::Convert::ToInt16(Console::ReadLine());    // Citirea noii
dimensiuni m
    for (j=0;j < m; j++){                                // Atribuire valori
aleatoare intre 0-100
        for (i=0;i < n; i++)
            numere[i,j]=nr->Next(100);
    }
    Console::WriteLine("\n\tNumerele sunt: \n\n");    //
Afisare numere
    for (j=0;j < m; j++){
        for (i=0;i < n; i++)
            Console::Write("  Nr["+i+", "+j+"]="+numere[i,j]+"\\t");
        Console::WriteLine(" ");
    }
    Console::ReadLine();
    return 0;
}

```



Sa realizam acum o aplicatie CLR Windows Forms Application care foloseste tablouri bidimensionale. Sa incercam sa desenam in mod continuu cercuri la diferite pozitii, de dimensiuni aleatoare si totodata sa stergem cercurile mai vechi in asa fel incat pe ecran sa avem un numar stabil de cercuri.

Deschidem un nou proiect Windows Forms Application intitulat **matr_sys** pe care plasam un obiect timer caruia ii setam intervalul la 10.

Vom utiliza o matrice care sa tina pozitiile x si y ale cercurilor precum si razele acestora. Pe fiecare eveniment Tick se va desena un cerc si se va sterge altul. Avand sa zicem 100 de cercuri incepem sa stergem cele mai vechi de 50 si vom avea in permanenta 50 de cercuri.

Pozitiile cercurilor si razele acestora se vor genera aleator in asa fel incat sa poata fi desenate pe form-ul curent. Sunt o serie de variabile si obiecte care trebuie sa aiba deci domeniul de vizibilitate in diverse proceduri deci trebuie puse in zona #pragma region .

Completam deci #pragma region cu :

C++

```
static int nr_c=100;
static int i=0;
static int j=0;
static int k=nr_c/2;
static System::Drawing::Graphics^ Desen;
static System::Drawing::Pen^ Creion_rosu ;
static System::Drawing::Pen^ Creion_pic ;
static System::Random^ n;
```

```
static array < int,2 >^ cercuri = gcnew array < int,2 > (nr_c,3);
```

C##

```
static int nr_c=100;
static int i=0;
static int j=0;
static int k=nr_c/2;
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_rosu ;
static System.Drawing.Pen Creion_pic ;
static System.Random n;
static int[,] cercuri = new int[nr_c, 3];
```

Completam procedura deschisa pe evenimentul Paint al obiectului form-ului cu :

C++

```
Desen = this->CreateGraphics();
Creion_rosu=gcnew System::Drawing::Pen(System::Drawing::Color::Red);
Creion_pic=gcnew System::Drawing::Pen(Color(this->BackColor));
n = gcnew System::Random();
Desen->Clear(System::Drawing::Color(this->BackColor));
for(i=0 ; i < nr_c ; i++){
    cercuri[i,0]=n->Next(this->Width);
    cercuri[i,1]=n->Next(this->Width);
    cercuri[i,2]=n->Next(75);
}
```

C#

```
Desen = this.CreateGraphics();
Creion_rosu=new System.Drawing.Pen(System.Drawing.Color.Red);
Creion_pic=new System.Drawing.Pen(this.BackColor);
n = new System.Random();
Desen.Clear(this.BackColor);
for(i=0 ; i < nr_c ; i++){
    cercuri[i,0]=n.Next(this.Width);
    cercuri[i,1]=n.Next(this.Width);
    cercuri[i,2]=n.Next(75);
}
```

```
}
```

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

C++

```
Desen->DrawEllipse( Creion_rosu, cercuri[j,0], cercuri[j,1],  
cercuri[j,2], cercuri[j,2]);  
Desen->DrawEllipse( Creion_pic, cercuri[k,0], cercuri[k,1],  
cercuri[k,2], cercuri[k,2]);  
j++;  
if (j > nr_c-1)  
    j=0;  
k++;  
if (k > nr_c-1)  
    k=0;
```

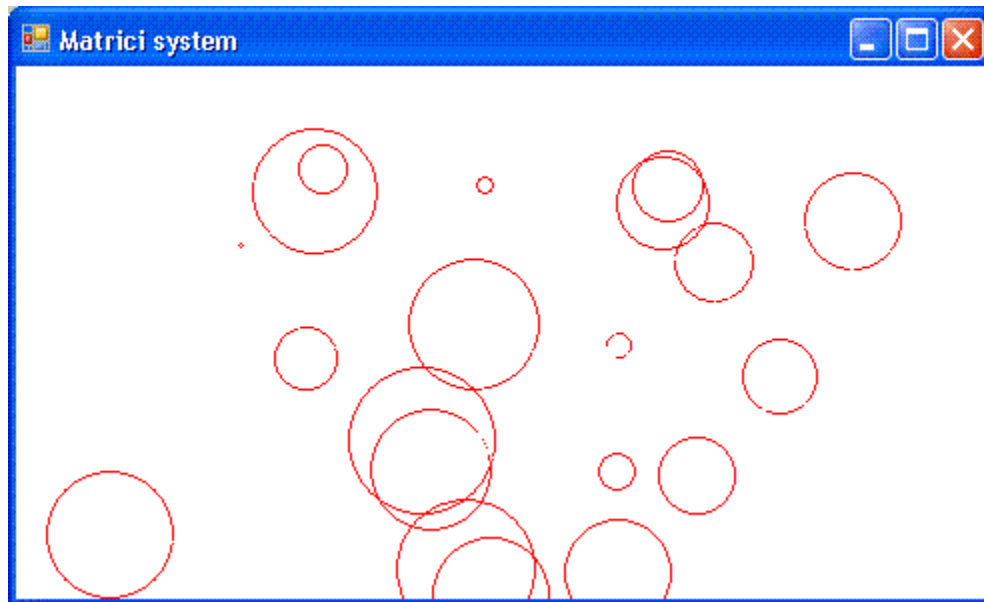
C#

```
Desen.DrawEllipse( Creion_rosu, cercuri[j,0], cercuri[j,1],  
cercuri[j,2], cercuri[j,2]);  
Desen.DrawEllipse( Creion_pic, cercuri[k,0], cercuri[k,1],  
cercuri[k,2], cercuri[k,2]);  
j++;  
if (j > nr_c-1)  
    j=0;  
k++;  
if (k > nr_c-1)  
    k=0;
```

Pentru C++ ompletam procedura deschisa pe evenimentul Deactivate al obiectului form-ului cu :

```
delete Desen;  
delete Creion_rosu;  
delete n;  
delete cercuri;
```

Dupa lansarea aplicatiei vom avea tot timpul 50 de cercuri pe ecran dar in diverse pozitii si de diverse marimi.



Se observa ca aparitia cercurilor nu este total aleatoare. Dupa o desenare a tuturor cercurilor se reia desenarea acelorasi cercuri. Pentru a evita repetarea schemei de cercuri, matricea cu coordonatele cercurilor trebuie regenerata. Nu le putem regenera pe toate pentru ca inca nu au fost sterse toate. Stergerea acestora este in urma cu $nr_c/2$, deci vom regenera numai prima jumatate de cercuri dupa ce s-a terminat de desenat cercurile si urmatoarea jumatate va fi generata numai dupa stergerea completa a primului set. Cu alte cuvinte problema este rezolvata prin rescrierea procedurii corespunzatoare evenimentului Tick al obiectului timer1 astfel:

C++

```
Desen->DrawEllipse( Creion_rosu, cercuri[j,0], cercuri[j,1],
cercuri[j,2], cercuri[j,2]);
Desen->DrawEllipse( Creion_pic, cercuri[k,0], cercuri[k,1],
cercuri[k,2], cercuri[k,2]);
j++;
if (j > nr_c-1){
    j=0;
    for(i=0 ; i < nr_c/2 ; i++){
        cercuri[i,0]=n->Next(this->Width);
        cercuri[i,1]=n->Next(this->Width);
        cercuri[i,2]=n->Next(75);
    }
}
k++;
if (k > nr_c-1)
    k=0;
```



```

        for(i=nr_c/2 ; i < nr_c ; i++){
            cercuri[i,0]=n->Next(this->Width);
            cercuri[i,1]=n->Next(this->Width);
            cercuri[i,2]=n->Next(75);
        }
    }

```

C#

```

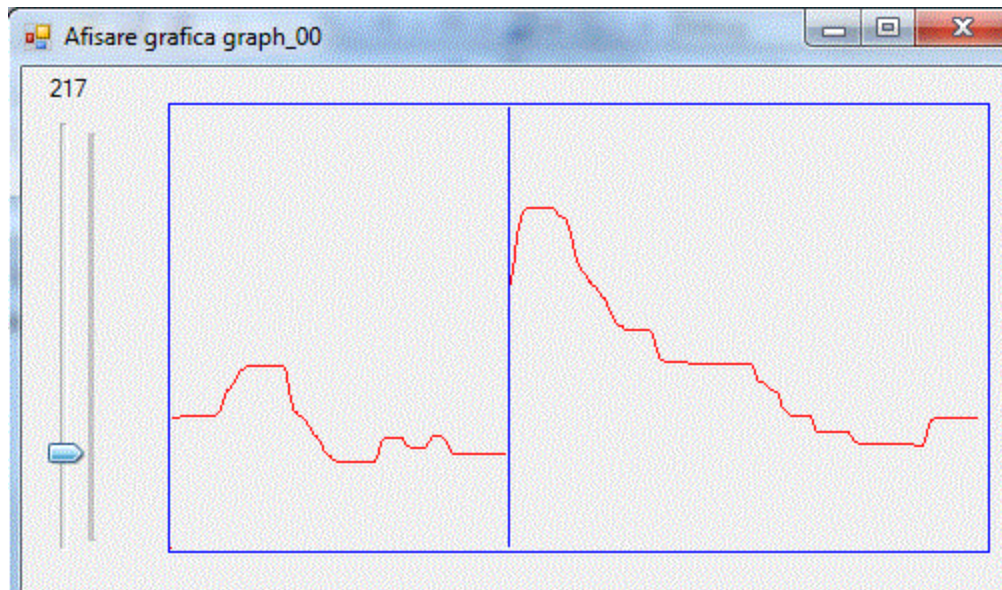
        Desen.DrawEllipse( Creion_rosu, cercuri[j,0], cercuri[j,1],
cercuri[j,2], cercuri[j,2]);
        Desen.DrawEllipse( Creion_pic, cercuri[k,0], cercuri[k,1],
cercuri[k,2], cercuri[k,2]);
        j++;
        if (j > nr_c-1){
            j=0;
            for(i=0 ; i < nr_c/2 ; i++){
                cercuri[i,0]=n.Next(this.Width);
                cercuri[i,1]=n.Next(this.Width);
                cercuri[i,2]=n.Next(75);
            }
        }
        k++;
        if (k > nr_c-1)
            k=0;
            for(i=nr_c/2 ; i < nr_c ; i++){
                cercuri[i,0]=n.Next(this.Width);
                cercuri[i,1]=n.Next(this.Width);
                cercuri[i,2]=n.Next(75);
            }
    }

```

Utilizarea tablourilor

Tablourile sunt des utilizate in achizitia si prelucrarea datelor provenite din monitorizarea diverselor procese. Vom incerca in continuare sa simulam date furnizate de sistemele de achizitie in vederea afisarii si prelucrarii acestora. Simularea datelor are avantajul ca nu necesita efectiv sisteme complexe de achizitie, totodata permitand punerea la punct a aplicatiilor care prelucreaza si afiseaza in forme diverse aceste date. Am realizat anterior cateva aplicatii care simulau date prin generarea aleatoare a acestora prin mecanisme random(). Aceste date astfel simulate nu se aseamana prea mult cu datele provenite din sistemele de achizitie. Vom folosi mouse-ul pentru a simula achizitia de date.

Pornim de la aplicatia "graph_00".



Vom crea aplicatia "achiz_v0" care foloseste un obiect trackBar pentru a genera date in vederea stocarii acestora in tablouri pentru a permite prelucrari complexe si afisari ale acestora.

Deschidem un nou proiect Windows Forms Application intitulat "achiz_v0" pe care plasam urmatoarele obiecte :

- un obiect de tip trackBar numit trackBar1
- un obiect de tip timer numit timer1 pe a carui eveniment tick se va face preluarea unei valori

C#

```
namespace achiz_v0
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        static int nr = 1;
        static int stare = 0;
        static int i = 0;
        static int scr_h ;
        static int val;
        static double scala;
        static int y, y_v;
    }
}
```

```

static int[] date_ac;
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_blu;
static System.Drawing.Pen Creion_red;
static System.Drawing.Pen Creion_pic;

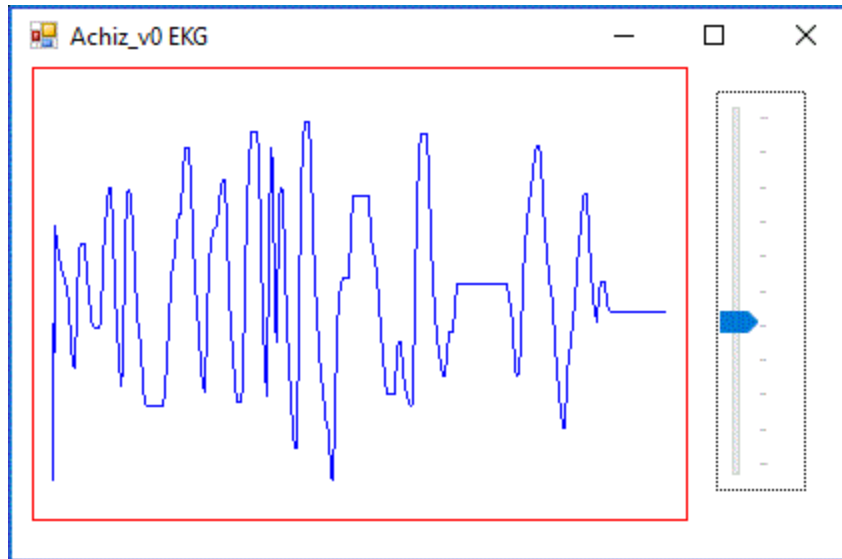
private void Form1_Load(object sender, EventArgs e)
{
    nr = this.Size.Width-110;
    scr_h = Screen.PrimaryScreen.Bounds.Height;
    stare = 1;
    i = 0;
    date_ac = new int[nr];
    Desen = this.CreateGraphics();
    Creion_blu = new System.Drawing.Pen(System.Drawing.Color.Blue);
    Creion_red = new System.Drawing.Pen(System.Drawing.Color.Red);
    Creion_pic = new System.Drawing.Pen(this.BackColor);

    scala = System.Convert.ToDouble(this.Size.Height - 80) /
this.trackBar1.Maximum;
}

private void timer1_Tick(object sender, EventArgs e)
{
    Desen.DrawRectangle(Creion_red, 10, 0, nr+1, this.Height-60);
    val = this.trackBar1.Value;
    for (i = 10; i < nr - 1; i++)
    {
        date_ac[i] = date_ac[i + 1];
    }
    date_ac[nr - 1] = val;

    y_v = System.Convert.ToInt16(this.Size.Height - 80 - scala *
date_ac[0]);
    for (i = 20; i < nr; i++)
    {
        y = System.Convert.ToInt16(this.Size.Height - 80 - scala *
date_ac[i]);
        Desen.DrawLine(Creion_pic, i+1,1, i + 1, this.Height-80);
        Desen.DrawLine(Creion_blu, i, y_v, i+1, y);
        y_v = y;
    }
}
}

```



Urmatoarea aplicatie foloseste mouse-ul pentru a genera date in vederea stocarii acestora in tabele pentru a permite prelucrari complexe si afisari ale acestora.

Deschidem un nou proiect Windows Forms Application intitulat "achiz_v1" pe care plasam urmatoarele obiecte :

- un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start"
- un obiect de tip label numit label1 utilizat pentru diverse mesaje
- un obiect de tip label numit label2 utilizat pentru afisarea valorii curente preluate
- un obiect de tip label numit label3 utilizat pentru afisarea indexului valorii preluate
- un obiect de tip timer numit timer1 pe a carui eveniment tick se va face preluarea unei valori

Dupa apasarea butonului "Start" se incepe achizitia de valori, numarul acestora depinzand de latimea ferestrei principale.

Achizitionarea si afisarea valorilor se face pe parcursul achizitionarii acestora. Dupa atingerea numarului maxim, achizitia este oprita. Fiecare valoare este obtinuta pe un eveniment tick al timer-ului. Procedura scrisa pe evenimentul click trebuie sa comunice procedurii scrise pe evenimentul tick cand sa inceapa achizitia prin setarea variabilei stare. Sunt o serie de variabile care trebuie sa aiba deci domeniul de vizibilitate in ambele proceduri deci trebuie puse in zona #pragma region .

Completam deci #pragma region cu :

C++

```
static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=1;
```

```

static int val;
static double scala;
static int y,y_v;
static System::Drawing::Graphics^ Desen;
static System::Drawing::Pen^ Creion_blu ;

```

C#

```

static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=1;
static int val;
static double scala;
static int y,y_v;
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_blu ;

```

Completam procedura deschisa pe evenimentul Click al obiectului button1 cu :

C++

```

        nr=this->Size.Width;
        stare=1;
        i=0;
        Desen= this->CreateGraphics();
        Creion_blu=gcnew System::Drawing::Pen(System::Drawing::Color::Blue);
        Desen->Clear(System::Drawing::Color(this->BackColor));
        scala=System::Convert::ToDouble(this->Size.Height-
120)/Screen::PrimaryScreen->Bounds.Height;
        scr_h=Screen::PrimaryScreen->Bounds.Height;
        this->label1->Text ="Se achizitioneaza
"+System::Convert::ToString(nr)+" valori";

```

C#

```

        nr=this.Size.Width;
        stare=1;
        i=0;
        Desen= this.CreateGraphics();

```

```

        Creion_blu=new System.Drawing.Pen(System.Drawing.Color.Blue);
        Desen.Clear(this.BackColor);
        scala=System.Convert.ToDouble(this.Size.Height-
120)/Screen.PrimaryScreen.Bounds.Height;
        scr_h=Screen.PrimaryScreen.Bounds.Height;
        this.labell1.Text ="Se achizitioneaza "+System.Convert.ToString(nr)+"
valori";

```

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

C++

```

if ((stare==1) && (i < nr)){
    this->label3->Text ="["+System::Convert::ToString(i)+"]=";
    val=scr_h-Control::MousePosition.Y;
    this->label2->Text = String::Concat(val);
    y=this->Size.Height-70-scala*val;
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i++;
}
if (i >= nr){
    this->labell1->Text ="Achizitie date terminata";
    stare=0;
    delete Desen;
    delete Creion_blu;
}

```

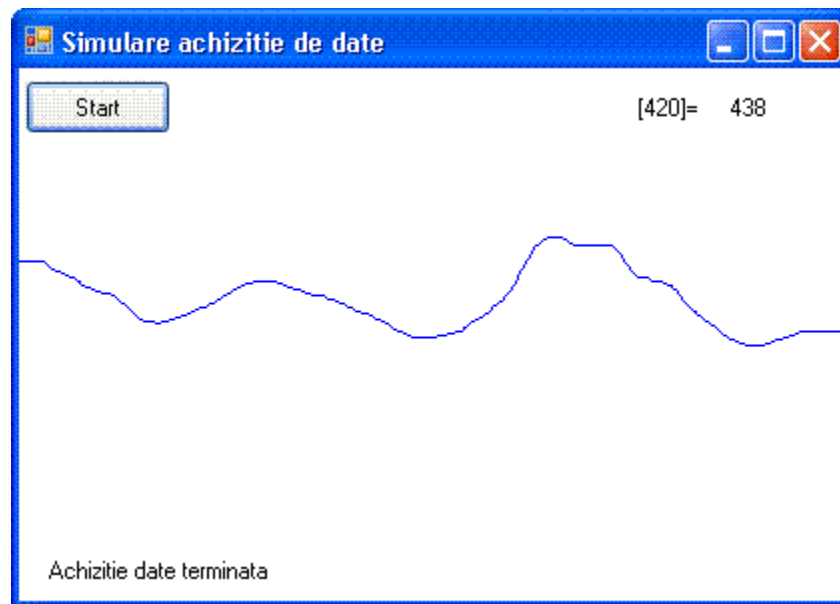
C#

```

if ((stare==1) && (i < nr)){
    this.label3.Text ="["+System.Convert.ToString(i)+"]=";
    val=scr_h-Control.MousePosition.Y;
    this.label2.Text = String.Concat(val);
    y = this.Size.Height - 70 - Convert.ToInt16(scala * val);
    Desen.DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i++;
}
if (i >= nr){
    this.labell1.Text ="Achizitie date terminata";
    stare=0;
}

```

Dupa apasarea butonului Start si miscarea mouse-ului pana se achizitioneaza toate valorile, se obtine:



Valorile astfel obtinute vor fi pastrate temporar in tabele in vederea prelucrarii acestora.

Pe baza aplicatiei anterioare vom adauga o serie de butoane cu care vom declansa diferite prelucrari ale datelor stocate in tabele.

Deschidem un nou proiect Windows Forms Application intitulat "achiz_v2" pe care plasam urmatoarele obiecte :

- un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start"
- un obiect de tip button numit button2 caruia ii schimbam atributul text in "Afisare date" si atributul Enabled in false
- un obiect de tip button numit button3 caruia ii schimbam atributul text in "Maxim" si atributul Enabled in false
- un obiect de tip button numit button4 caruia ii schimbam atributul text in "Minim" si atributul Enabled in false
- un obiect de tip button numit button5 caruia ii schimbam atributul text in "Med" si atributul Enabled in false
- un obiect de tip label numit label1 utilizat pentru diverse mesaje
- un obiect de tip label numit label2 utilizat pentru afisarea valorii curente preluate
- un obiect de tip label numit label3 utilizat pentru afisarea indexului valorii preluate
- un obiect de tip timer numit timer1 pe a carui eveniment tick se va face preluarea unei valori

Completam deci #pragma region cu :

C++

```

static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=Screen::PrimaryScreen->Bounds.Height;
static int val;
static double scala;
static int y,y_v;
static array < int >^ date_ac = gcnew array < int > (0);
static System::Drawing::Graphics^ Desen;
static System::Drawing::Pen^ Creion_blu ;
static System::Drawing::Pen^ Creion_red ;
static System::Drawing::SolidBrush^ Pensula;
static System::Drawing::Font^ font_nina;

int maxim(array < int >^ date, int nrm){
    int max=date[0];
    for (int i=0;i < nrm;i++){
        if (max < date[i])
            max=date[i];
    }
    return max;
}
int minim(array < int >^ date, int nrm){
    int min=date[0];
    for (int i=0;i < nrm;i++){
        if (min > date[i])
            min=date[i];
    }
    return min;
}
double med(array < int >^ date, int nrm){
    double med=0;
    for (int i=0;i < nrm;i++){
        med=med+System::Convert::ToDouble(date[i])/nrm;
    }
    return med;
}

```

C#

```

static int nr = 1;
static int stare = 0;
static int i = 0;
static int scr_h = Screen.PrimaryScreen.Bounds.Height;
static int val;
static double scala;
static int factor = 0;

```



```

static int y, y_v;
static int[] date_ac = new int[0];
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_blu;
static System.Drawing.Pen Creion_red;
static System.Drawing.SolidBrush Pensula;
static System.Drawing.Font font_nina;
int maxim(int[] date, int nrm)
{
    int max = date[0];
    for (int i = 0; i < nrm; i++)
    {
        if (max < date[i])
            max = date[i];
    }
    return max;
}
int minim(int[] date, int nrm)
{
    int min = date[0];
    for (int i = 0; i < nrm; i++)
    {
        if (min > date[i])
            min = date[i];
    }
    return min;
}
double med(int[] date, int nrm)
{
    double med = 0;
    for (int i = 0; i < nrm; i++)
    {
        med = med + System.Convert.ToDouble(date[i]) / nrm;
    }
    return med;
}

```

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

C++

```

if ((stare==1) && (i < nr)){
    this->button2->Enabled=false;
    this->button3->Enabled=false;
    this->button4->Enabled=false;
    this->button5->Enabled=false;
    this->label3->Text = "["+System.Convert.ToString(i)+"]=";
    val=scr_h-Control::MousePosition.Y;
    this->label2->Text = String::Concat(val);
    date_ac[i]=val;
    y=this->Size.Height-70-scala*val;
}

```

```

        Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
        y_v=y;
        i++;
    }
    if (i >= nr){
        this->labell1->Text ="Achizitie date terminata";
        stare=0;
        this->button2->Enabled=true;
        this->button3->Enabled=true;
        this->button4->Enabled=true;
        this->button5->Enabled=true;
    }

```

C#

```

        if ((stare == 1) && (i < nr))
        {
            this.button2.Enabled = false;
            this.button3.Enabled = false;
            this.button4.Enabled = false;
            this.button5.Enabled = false;
            this.label3.Text = "[" + System.Convert.ToString(i) + "]=";
            val = scr_h - Control.MousePosition.Y;
            this.label2.Text = String.Concat(val);
            date_ac[i] = val;
            y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
val);

            Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
            y_v = y;
            i++;
        }
        if (i >= nr)
        {
            this.labell1.Text = "Achizitie date terminata";
            stare = 0;
            this.button2.Enabled = true;
            this.button3.Enabled = true;
            this.button4.Enabled = true;
            this.button5.Enabled = true;
        }

```

Completam procedura deschisa pe evenimentul Click al obiectului button1 cu :

C++

```

        nr=this->Size.Width;
        Array::Resize(date_ac,nr);
        stare=1;
        i=0;
        Desen= this->CreateGraphics();
        Creion_blu=gcnew System::Drawing::Pen(System::Drawing::Color::Blue);
        Creion_red=gcnew System::Drawing::Pen(System::Drawing::Color::Red);
        scala=System::Convert::ToDouble(this->Size.Height-
120)/Screen::PrimaryScreen->Bounds.Height;
        Pensula=gcnew
System::Drawing::SolidBrush(System::Drawing::Color::DarkOrchid );
        font_nina=gcnew System::Drawing::Font("Nina",8);
        Desen->Clear(System::Drawing::Color(this->BackColor));
        this->labell->Text ="Se achizitioneaza
"+System::Convert::ToString(nr)+" valori";
        y=this->Size.Height-70-scala*(scr_h-Control::MousePosition.Y);

```

C#

```

        nr = this.Size.Width;
        factor = date_ac.Length;
        Array.Resize(ref date_ac, nr);
        stare = 1;
        i = 0;
        Desen = this.CreateGraphics();
        Creion_blu = new System.Drawing.Pen(System.Drawing.Color.Blue);
        Creion_red = new System.Drawing.Pen(System.Drawing.Color.Red);
        scala = System.Convert.ToDouble(this.Size.Height - 120) /
Screen.PrimaryScreen.Bounds.Height;
        Pensula = new
System.Drawing.SolidBrush(System.Drawing.Color.DarkOrchid);
        font_nina = new System.Drawing.Font("Nina", 8);
        Desen.Clear(this.BackColor);
        this.labell.Text = "Se achizitioneaza " +
System.Convert.ToString(nr) + " valori";
        y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
(scr_h - Control.MousePosition.Y));

```

Completam procedura deschisa pe evenimentul Click al obiectului button2 cu :

C++

```

Desen->Clear(System::Drawing::Color(this->BackColor));
int r=1,j=1;

```

```

        i=0;
        for ( int i=0; i < nr; i++){
            Desen-
>DrawString(System::Convert::ToString(date_ac[i]),font_nina,Pensula,10+j*35,30+11*r);

            j++;
            if (j==11){
                r++;
                j=1;
            }
            if (i > 198) i=nr;// limitez la 198 de valori
        }
        this->label1->Text ="Primele 200 de valori";

```

C#

```

        Desen.Clear(this.BackColor);
        int r = 1, j = 1, i = 0;
        for (i = 0; i < nr; i++)
        {
            Desen.DrawString(System.Convert.ToString(date_ac[i]),
font_nina, Pensula, 10 + j * 35, 30 + 11 * r);
            j++;
            if (j == 11)
            {
                r++;
                j = 1;
            }
            if (i > 198) i = nr;// limitez la 198 de valori
        }
        this.label1.Text = "Primele 200 de valori";

```

Completam procedura deschisa pe evenimentul Click al obiectului button3 cu :

C++

```

this->label3->Text = "V_max=";
this->label2->Text = System::Convert::ToString(maxim(date_ac,nr));
Desen->Clear(System::Drawing::Color(this->BackColor));
i=0;
do {
    y=this->Size.Height-70-scala*date_ac[i];
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i+=1;
}

```

```

    }
    while (i < nr);
        y=this->Size.Height-70-scala*(maxim(date_ac,nr));
        Desen->DrawLine(Creion_red,0,y, nr, y);

```

C#

```

        this.label3.Text = "V_max=";
        this.label2.Text = System.Convert.ToString(maxim(date_ac, nr));
        Desen.Clear(this.BackColor);
        i = 0;
        do
        {
            y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
date_ac[i]);
            Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
            y_v = y;
            i += 1;
        }
        while (i < nr);
        y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
(maxim(date_ac, nr)));
        Desen.DrawLine(Creion_red, 0, y, nr, y);

```

Completam procedura deschisa pe evenimentul Click al obiectului button4 cu :

C++

```

this->label3->Text = "V_min=";
this->label2->Text = System::Convert::ToString(minim(date_ac,nr));

Desen->Clear(System::Drawing::Color(this->BackColor));
i=0;
do {
    y=this->Size.Height-70-scala*date_ac[i];
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i+=1;
}
while (i < nr);
    y=this->Size.Height-70-scala*(minim(date_ac,nr));
    Desen->DrawLine(Creion_red,0,y, nr, y);

```

C#

```
        this.label3.Text = "V_min=";
        this.label2.Text = System.Convert.ToString(minim(date_ac, nr));

        Desen.Clear(this.BackColor);
        i = 0;
        do
        {
            y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
date_ac[i]);
            Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
            y_v = y;
            i += 1;
        }
        while (i < nr);
        y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
(minim(date_ac, nr)));
        Desen.DrawLine(Creion_red, 0, y, nr, y);
```

Completam procedura deschisa pe evenimentul Click al obiectului button5 cu :

C++

```
this->label3->Text = "V_med=";
this->label2->Text = System::Convert::ToString(med(date_ac,nr));
Desen->Clear(System::Drawing::Color(this->BackColor));
i=0;
do {
    y=this->Size.Height-70-scala*date_ac[i];
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i+=1;
}
while (i < nr);
y=this->Size.Height-70-scala*(med(date_ac,nr));
Desen->DrawLine(Creion_red,0,y, nr, y);
```

C#

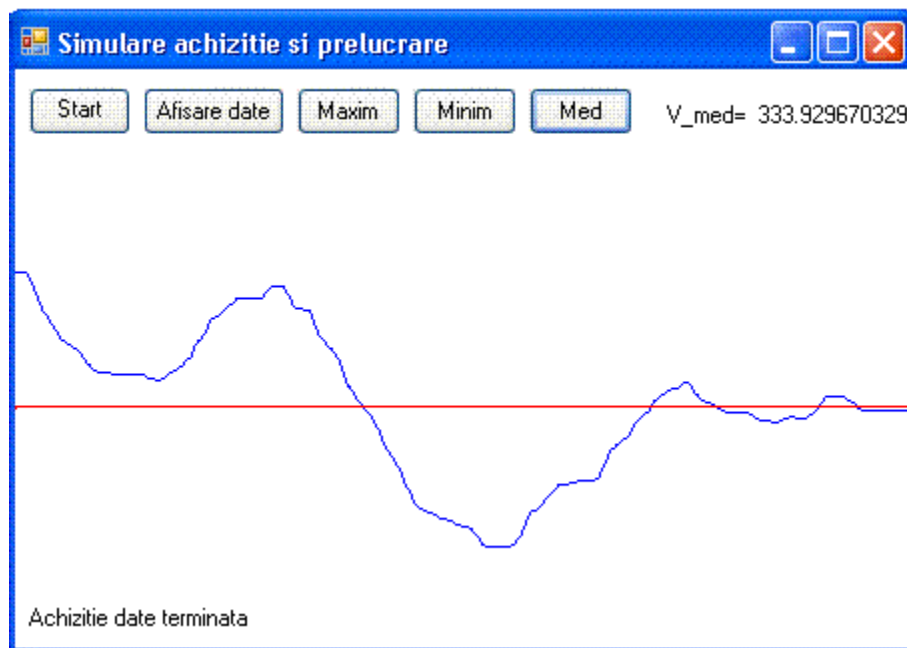
```
this.label3.Text = "V_med=";
this.label2.Text = System.Convert.ToString(med(date_ac, nr));
Desen.Clear(this.BackColor);
```

```

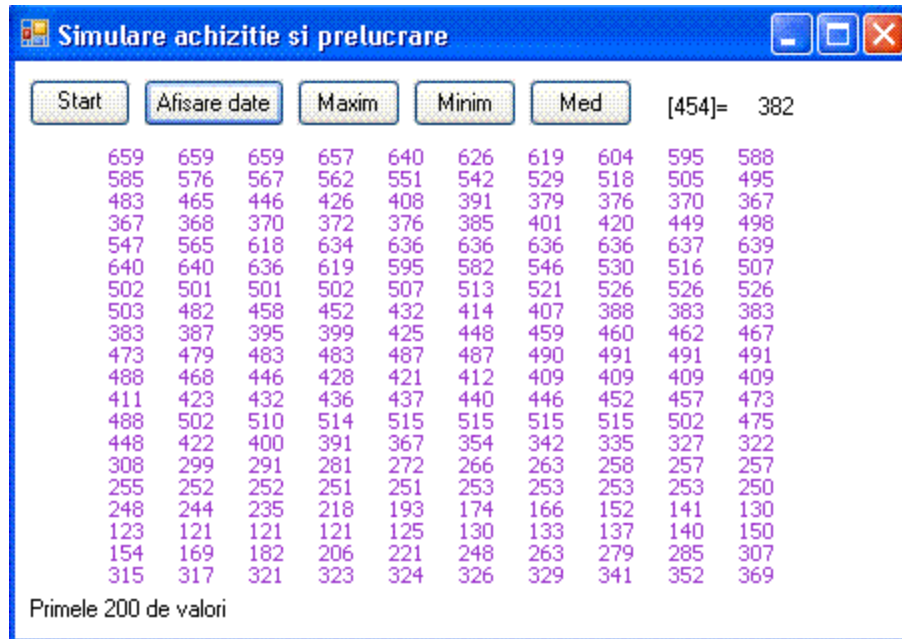
        i = 0;
        do
        {
            y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
date_ac[i]);
            Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
            y_v = y;
            i += 1;
        }
        while (i < nr);
        y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
(med(date_ac, nr)));
        Desen.DrawLine(Creion_red, 0, y, nr, y);

```

Dupa apasarea butonului Start, miscarea mouse-ului pana se achizitioneaza toate valorile si apasarea butonului "Med", se obtine:



Daca dupa apasarea butonului Start, miscarea mouse-ului pana se achizitioneaza toate valorile si apasarea butonului "Afisare date", se obtine:



Pe parcursul achizitiei de date valoarea maxima nu se cunoaste, aceasta putandu-se determina numai dupa ce toate datele au fost achizitionate. Este de preferat sa rescalam graficul functie de valoarea maxima atinsa astfel incat graficul sa cuprinda intreaga suprafata destinata afisarii grafice a datelor. Urmatoarea aplicatie permite rescalarea datelor si introducerea efectului de lupa in vederea analizei formei de unda a semnalului achizitionat. Se vor introduce deci butoane pentru afisarea scalata si butoane pentru "amplificarea" semnalului.

Deschidem un nou proiect Windows Forms Application intitulat "achiz_v3" pe care plasam urmatoarele obiecte :

- un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start"
- un obiect de tip button numit button2 caruia ii schimbam atributul text in "Afisare scalata" si atributul Enabled in false
- un obiect de tip button numit button3 caruia ii schimbam atributul text in "+" si atributul Enabled in false
- un obiect de tip button numit button4 caruia ii schimbam atributul text in "++" si atributul Enabled in false
- un obiect de tip label numit label1 utilizat pentru diverse mesaje
- un obiect de tip label numit label2 utilizat pentru afisarea valorii curente preluata
- un obiect de tip label numit label3 utilizat pentru afisarea indexului valorii preluate
- un obiect de tip timer numit timer1 pe a carui eveniment tick se va face preluarea unei valori

Completam deci #pragma region cu :

C++


```

static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=Screen::PrimaryScreen->Bounds.Height;
static int val;
static double scala;
static int factor=0;
static int y,y_v;
static array < int >^ date_ac = gcnew array < int > (0);
static System::Drawing::Graphics^ Desen;
static System::Drawing::Pen^ Creion_blu ;

int maxim(array < int >^ date, int nrm){
    int max=date[0];
    for (int i=0;i < nrm;i++){
        if (max < date[i])
            max=date[i];
    }
    return max;
}
int minim(array < int >^ date, int nrm){
    int min=date[0];
    for (int i=0;i < nrm;i++){
        if (min > date[i])
            min=date[i];
    }
    return min;
}

```

C#

```

static int nr = 1;
static int stare = 0;
static int i = 0;
static int scr_h = Screen.PrimaryScreen.Bounds.Height;
static int val;
static double scala;
static int factor = 0;
static int y, y_v;
static int[] date_ac = new int[0];
static System.Drawing.Graphics Desen;
static System.Drawing.Pen Creion_blu;

int maxim(int[] date, int nrm)
{
    int max = date[0];
    for (int i = 0; i < nrm; i++)

```

```

        {
            if (max < date[i])
                max = date[i];
        }
        return max;
    }
    int minim(int[] date, int nrm)
    {
        int min = date[0];
        for (int i = 0; i < nrm; i++)
        {
            if (min > date[i])
                min = date[i];
        }
        return min;
    }
}

```

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

C++

```

if ((stare==1) && (i < nr)){
    this->button2->Enabled=false;
    this->button3->Enabled=false;
    this->button4->Enabled=false;
    this->label3->Text = "["+System::Convert::ToString(i)+"]=";
    val=scr_h-Control::MousePosition.Y;
    this->label2->Text = String::Concat(val);
    date_ac[i]=val;
    y=this->Size.Height-70-scala*val;
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i++;
}
if (i >= nr){
    this->label1->Text = "Achizitie date terminata";
    stare=0;
    this->button2->Enabled=true;
    this->button3->Enabled=true;
    this->button4->Enabled=true;
}
}

```

C#

```

if ((stare == 1) && (i < nr))

```

```

        {
            this.button2.Enabled = false;
            this.button3.Enabled = false;
            this.button4.Enabled = false;
            this.label3.Text = "[" + System.Convert.ToString(i) + "]=";
            val = scr_h - Control.MousePosition.Y;
            this.label2.Text = String.Concat(val);
            date_ac[i] = val;
            y = System.Convert.ToInt16(this.Size.Height - 70 - scala *
val);

            Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
            y_v = y;
            i++;
        }
        if (i >= nr)
        {
            this.label1.Text = "Achizitie date terminata";
            stare = 0;
            this.button2.Enabled = true;
            this.button3.Enabled = true;
            this.button4.Enabled = true;
        }
    }

```

Completam procedura deschisa pe evenimentul Click al obiectului button1 cu :

C++

```

Desen= this->CreateGraphics();
Creion_blu=gcnw System::Drawing::Pen(System::Drawing::Color::Blue);
scala=System::Convert::ToDouble(this->Size.Height-
120)/Screen::PrimaryScreen->Bounds.Height;
nr=this->Size.Width;
Array::Resize(date_ac,nr);
stare=1;
factor=0;
i=0;
Desen->Clear(System::Drawing::Color(this->BackColor));
this->label1->Text ="Se achizitioneaza
"+System::Convert::ToString(nr)+" valori";
y=this->Size.Height-70-scala*(scr_h-Control::MousePosition.Y);

```

C#

```

Desen = this.CreateGraphics();
Creion_blu = new System.Drawing.Pen(System.Drawing.Color.Blue);

```

```

        scala = System.Convert.ToDouble(this.Size.Height - 120) /
Screen.PrimaryScreen.Bounds.Height;
        nr = this.Size.Width;
        Array.Resize(ref date_ac, nr);

        factor = 0;
        i = 0;
        Desen.Clear(this.BackColor);
        this.labell.Text = "Se achizitioneaza " +
System.Convert.ToString(nr) + " valori";
        y = 50+System.Convert.ToInt16(this.Size.Height - 70 - scala *
(scr_h - Control.MousePosition.Y));
        y_v = y;
        stare = 1;

```

Completam procedura deschisa pe evenimentul Click al obiectului button2 cu :

C++

```

Desen->Clear(System::Drawing::Color(this->BackColor));
double scala_m=System::Convert::ToDouble(this->Size.Height-
120)/(maxim(date_ac,nr));
i=0;
do {
    y=this->Size.Height-70-scala_m*(date_ac[i]);
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i+=1;
}
while (i < nr);

```

C#

```

Desen.Clear(this.BackColor);
double scala_m = System.Convert.ToDouble(this.Size.Height - 120)
/ (maxim(date_ac, nr));
i = 0;
do
{
    y = System.Convert.ToInt16(this.Size.Height - 70 - scala_m *
(date_ac[i]));
    Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
    y_v = y;
    i += 1;
}

```

```
while (i < nr);
```

Completam procedura deschisa pe evenimentul Click al obiectului button3 cu :

C++

```
if (factor < minim(date_ac,nr)-40)
factor+=40;
Desen->Clear(System::Drawing::Color(this->BackColor));
double scala_m=System::Convert::ToDouble(this->Size.Height-
120)/(maxim(date_ac,nr)-factor);
i=0;
do {
    y=this->Size.Height-70-scala_m*(date_ac[i]-factor);
    Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
    y_v=y;
    i+=1;
}
while (i < nr);
```

C#

```
if (factor < minim(date_ac, nr) - 40)
    factor += 40;
Desen.Clear(this.BackColor);
double scala_m = System.Convert.ToDouble(this.Size.Height - 120)
/ (maxim(date_ac, nr) - factor);
i = 0;
do
{
    y = System.Convert.ToInt16(this.Size.Height - 70 - scala_m *
(date_ac[i] - factor));
    Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
    y_v = y;
    i += 1;
}
while (i < nr);
```

Completam procedura deschisa pe evenimentul Click al obiectului button4 cu :

C++

```

        Desen->Clear(System::Drawing::Color(this->BackColor));
        double scala_m=System::Convert::ToDouble(this->Size.Height-
120)/(maxim(date_ac,nr)-
        minim(date_ac,nr));
        factor=0;
        i=0;
        do {
                y=this->Size.Height-70-scala_m*(date_ac[i]-
minim(date_ac,nr));
                Desen->DrawLine(Creion_blu,i-1,y_v, i, y);
                y_v=y;
                i+=1;
        }
        while (i < nr);

```

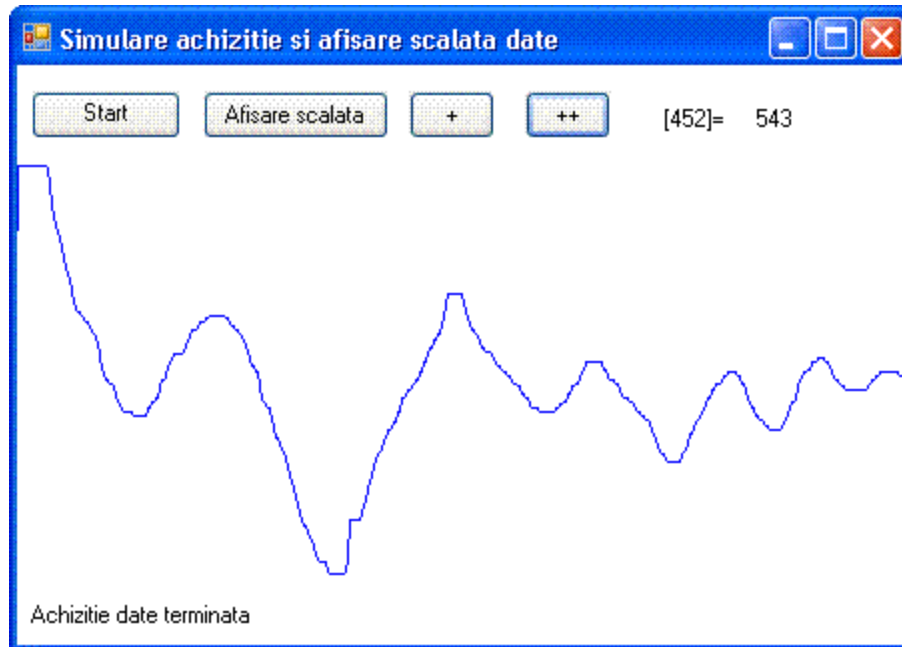
C#

```

        Desen.Clear(this.BackColor);
        double scala_m = System.Convert.ToDouble(this.Size.Height - 120)
/ (maxim(date_ac, nr) -
        minim(date_ac, nr));
        factor = 0;
        i = 0;
        do
        {
                y = System.Convert.ToInt16(this.Size.Height - 70 - scala_m *
(date_ac[i] - minim(date_ac, nr)));
                Desen.DrawLine(Creion_blu, i - 1, y_v, i, y);
                y_v = y;
                i += 1;
        }
        while (i < nr);

```

Dupa apasarea butonului Start, miscarea mouse-ului pana se achizitioneaza toate valorile si apasarea butonului "++", se obtine:



Sa incercam acum simularea a doua semnale folosind miscarea mouse-ului pe x respectiv pe y . Cele doua semnale ar putea reprezenta de exemplu tensiunea si curentul achizitionate dintr-un sistem electric.

Deschidem un nou proiect Windows Forms Application intitulat "achiz_v4" pe care plasam urmatoarele obiecte :

- un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start"
- un obiect de tip button numit button2 caruia ii schimbam atributul text in "Tensiune" si atributul Enabled in false
- un obiect de tip button numit button3 caruia ii schimbam atributul text in "Curent" si atributul Enabled in false
- un obiect de tip button numit button4 caruia ii schimbam atributul text in " $P=U \cdot I$ " si atributul Enabled in false
- un obiect de tip button numit button5 caruia ii schimbam atributul text in "Sterg" si atributul Enabled in false
- un obiect de tip label numit label1 utilizat pentru diverse mesaje
- un obiect de tip label numit label2 utilizat pentru afisarea valorii tensiunii
- un obiect de tip label numit label3 utilizat pentru afisarea indexului valorilor preluate
- un obiect de tip label numit label4 utilizat pentru afisarea valorii curentului
- un obiect de tip timer numit timer1 pe a carui eveniment tick se va face preluarea a doua valori (u,I)

Completam deci #pragma region cu :

C++

```

static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=Screen::PrimaryScreen->Bounds.Height;
static int scr_w=Screen::PrimaryScreen->Bounds.Width;
static double scala_h;
static double scala_w;
static int val_u;      // tensiune
static int val_c;      // curent
static int val_p;      // putere
static int u,u_v,c,c_v,p,p_v;
static array < int >^ tens = gcnew array < int > (0);
static array < int >^ crnt = gcnew array < int > (0);
static int factor;
static System::Drawing::Graphics^ Desen ;
static System::Drawing::Pen^ Creion_blu ;
static System::Drawing::Pen^ Creion_red ;
static System::Drawing::Pen^ Creion_green ;

int max_p(array < int >^ datel,array < int >^ date2, int nrm){
    int max=datel[0]*date2[0];
    for (int i=0;i < nrm;i++){
        if (max < datel[i]*date2[i])
            max=datel[i]*date2[i];
    }
    return max;
}

```

C#

```

static int nr=1;
static int stare=0;
static int i=0;
static int scr_h=Screen.PrimaryScreen.Bounds.Height;
static int scr_w=Screen.PrimaryScreen.Bounds.Width;
static double scala_h;
static double scala_w;
static int val_u;      // tensiune
static int val_c;      // curent
static int val_p;      // putere
static int u,u_v,c,c_v,p,p_v;
static int [] tens = new int[0];
static int [] crnt = new int[0];
static int factor;
static System.Drawing.Graphics Desen ;
static System.Drawing.Pen Creion_blu ;
static System.Drawing.Pen Creion_red ;
static System.Drawing.Pen Creion_green ;

```



```

int max_p(int[] date1, int[] date2, int nrm){
    int max=date1[0]*date2[0];
    for (int i=0;i < nrm;i++){
        if (max < date1[i]*date2[i])
            max=date1[i]*date2[i];
    }
    return max;
}

```

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

C++

```

if ((stare==1) && (i < nr)){
    this->button2->Enabled=false;
    this->button3->Enabled=false;
    this->button3->Enabled=false;
    this->button4->Enabled=false;
    this->button5->Enabled=false;
    this->label3->Text = "["+System::Convert::ToString(i)+"]=";
    val_u=scr_h-Control::MousePosition.Y;
    this->label2->Text = String::Concat(val_u);
    tens[i]=val_u;
    u=this->Size.Height-70-scala_h*val_u;
    Desen->DrawLine(Creion_blu,i-1,u_v, i, u);
    u_v=u;

    val_c=Control::MousePosition.X;
    this->label4->Text = String::Concat(val_c);
    crnt[i]=val_c;
    c=this->Size.Height-70-scala_w*val_c;
    Desen->DrawLine(Creion_red,i-1,c_v, i, c);
    c_v=c;
    i++;
}
if (i >= nr){
    this->label1->Text = "Achizitie date terminata";
    stare=0;
    this->button2->Enabled=true;
    this->button3->Enabled=true;
    this->button3->Enabled=true;
    this->button4->Enabled=true;
    this->button5->Enabled=true;
}

```

C#

```

if ((stare==1) && (i < nr)){
    this.button2.Enabled=false;
    this.button3.Enabled=false;
    this.button3.Enabled=false;
    this.button4.Enabled=false;
    this.button5.Enabled=false;
    this.label3.Text = "["+System.Convert.ToString(i)+"]=";
    val_u=scr_h-Control.MousePosition.Y;
    this.label2.Text = String.Concat(val_u);
    tens[i]=val_u;
    u=System.Convert.ToInt16(this.Size.Height-70-scala_h*val_u);
    Desen.DrawLine(Creion_blu,i-1,u_v, i, u);
    u_v=u;
    val_c=Control.MousePosition.X;
    this.label4.Text = String.Concat(val_c);
    crnt[i]=val_c;
    c=System.Convert.ToInt16(this.Size.Height-70-scala_w*val_c);
    Desen.DrawLine(Creion_red,i-1,c_v, i, c);
    c_v=c;
    i++;
}
if (i >= nr){
    this.label1.Text = "Achizitie date terminata";
    stare=0;
    this.button2.Enabled=true;
    this.button3.Enabled=true;
    this.button3.Enabled=true;
    this.button4.Enabled=true;
    this.button5.Enabled=true;
}

```

Completam procedura deschisa pe evenimentul Click al obiectului button1 cu :

C++

```

Desen= this->CreateGraphics();
Creion_blu=gcnew System::Drawing::Pen(System::Drawing::Color::Blue);
Creion_red=gcnew System::Drawing::Pen(System::Drawing::Color::Red);
Creion_green=gcnew
System::Drawing::Pen(System::Drawing::Color::Green);
scala_h=System::Convert::ToDouble(this->Size.Height-
120)/Screen::PrimaryScreen->Bounds.Height/2;
scala_w=System::Convert::ToDouble(this->Size.Height-
120)/Screen::PrimaryScreen->Bounds.Width/2;
nr=this->Size.Width;
Array::Resize(tens,nr);
Array::Resize(crnt,nr);
stare=1;
factor=0;
i=0;

```

```

        Desen->Clear(System::Drawing::Color(this->BackColor));
        this->label1->Text ="Se achizitioneaza
"+System::Convert::ToString(nr)+" valori";

```

C#

```

        Desen= this.CreateGraphics();
        Creion_blu=new System.Drawing.Pen(System.Drawing.Color.Blue);
        Creion_red=new System.Drawing.Pen(System.Drawing.Color.Red);
        Creion_green=new System.Drawing.Pen(System.Drawing.Color.Green);
        scala_h=System.Convert.ToDouble(this.Size.Height-
120)/Screen.PrimaryScreen.Bounds.Height/2;
        scala_w=System.Convert.ToDouble(this.Size.Height-
120)/Screen.PrimaryScreen.Bounds.Width/2;
        nr=this.Size.Width;
        Array.Resize(ref tens,nr);
        Array.Resize(ref crnt,nr);
        stare=1;
        factor=0;
        i=0;
        Desen.Clear(this.BackColor);
        this.label1.Text ="Se achizitioneaza "+System.Convert.ToString(nr)+"
valori";

```

Completam procedura deschisa pe evenimentul Click al obiectului button2 cu :

C++

```

        i=0;
        do {
            u=this->Size.Height-70-scala_h*(tens[i]-factor);
            Desen->DrawLine(Creion_blu,i-1,u_v, i, u);
            u_v=u;
            i+=1;
        }
        while (i < nr);

```

C#

```

        i=0;

```

```

do {
    u=System.Convert.ToInt16(this.Size.Height-70-
scala_h*(tens[i]-factor));
    Desen.DrawLine(Creion_blu,i-1,u_v, i, u);
    u_v=u;
    i+=1;
}
while (i < nr);

```

Completam procedura deschisa pe evenimentul Click al obiectului button3 cu :

C++

```

i=0;
do {
    c=this->Size.Height-70-scala_w*(crnt[i]-factor);
    Desen->DrawLine(Creion_red,i-1,c_v, i, c);
    c_v=c;
    i+=1;
}
while (i < nr);

```

C#

```

i=0;
do {
    c=System.Convert.ToInt16(this.Size.Height-70-
scala_w*(crnt[i]-factor));
    Desen.DrawLine(Creion_red,i-1,c_v, i, c);
    c_v=c;
    i+=1;
}
while (i < nr);

```

Completam procedura deschisa pe evenimentul Click al obiectului button4 cu :

C++

```

double scala_m=System::Convert::ToDouble(this->Size.Height-
120)/(max_p(tens,crnt,nr)-factor);

```

```

i=0;
do {
    p=this->Size.Height-70-scala_m*(tens[i]*crnt[i]-factor);
    Desen->DrawLine(Creion_green,i-1,p_v, i, p);
    p_v=p;
    i+=1;
}
while (i < nr);

```

C#

```

double scala_m=System.Convert.ToDouble(this.Size.Height-
120)/(max_p(tens,crnt,nr)-factor);
i=0;
do {
    p=System.Convert.ToInt16(this.Size.Height-70-
scala_m*(tens[i]*crnt[i]-factor));
    Desen.DrawLine(Creion_green,i-1,p_v, i, p);
    p_v=p;
    i+=1;
}
while (i < nr);

```

Completam procedura deschisa pe evenimentul Click al obiectului button5 cu :

C++

```

Desen->Clear(System::Drawing::Color(this->BackColor));

```

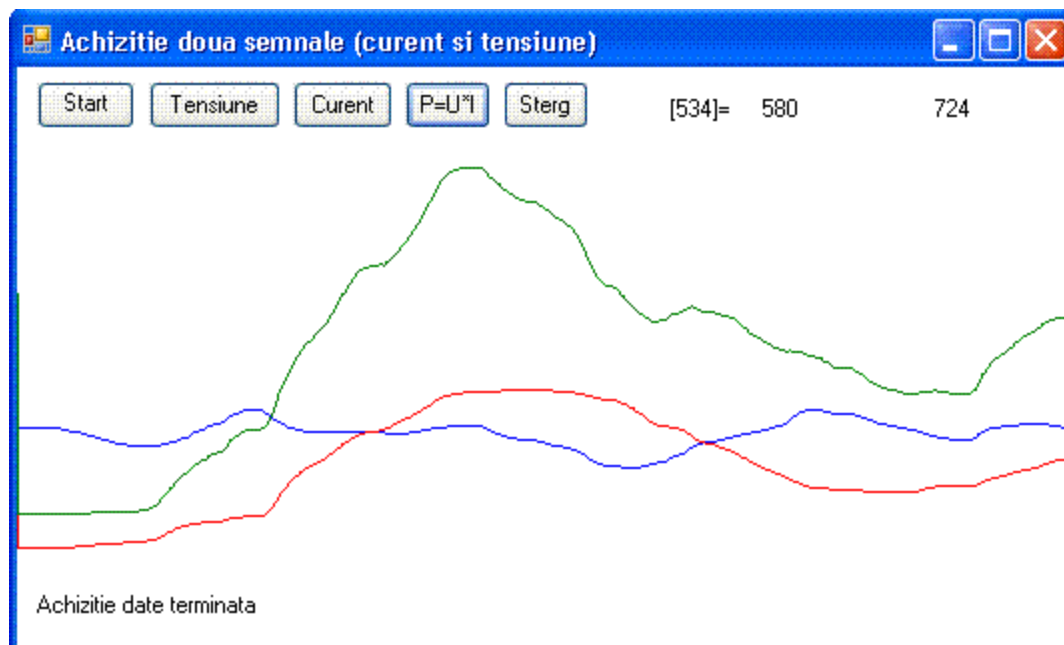
C#

```

Desen.Clear(this.BackColor);

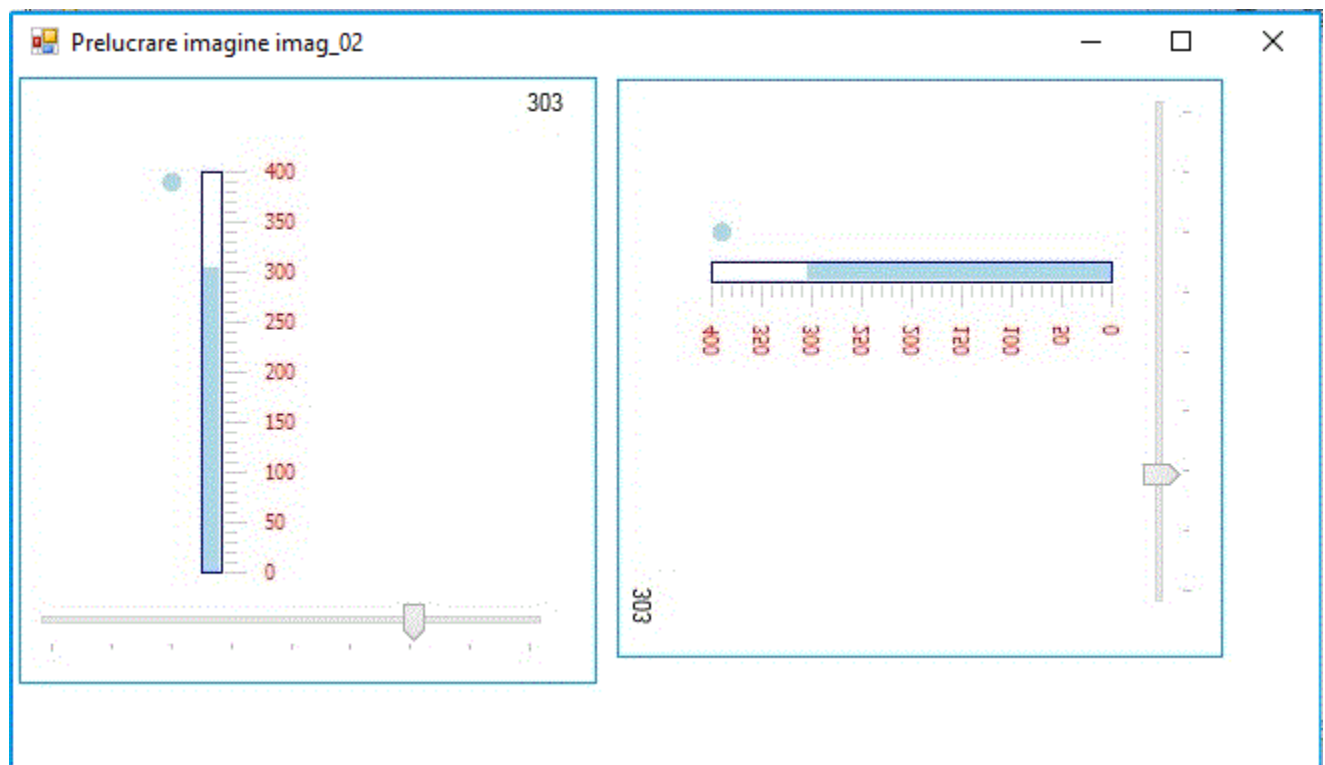
```

Dupa apasarea butonului Start, miscarea mouse-ului pe x si y pana se achizitioneaza toate valorile si apasarea butonului "P=U*I", se obtine:



Prelucrarea imaginilor

In urmatoarea aplicatie vom incerca sa incarcam o imagine, sa o afisam dupa care sa rotim stanga imaginea si sa o o afisam din nou.



Codul sursa in C# fiind:

```
namespace imag_02
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        System.Drawing.Bitmap img;
        System.Drawing.Bitmap img1;
        System.Drawing.Graphics Desen;
        System.Drawing.Color cul;
        private void Form1_Load(object sender, EventArgs e)
        {
            img=new Bitmap("termo.jpg");
            img1 = new Bitmap(309,295);
            Desen = this.CreateGraphics();
            cul = new System.Drawing.Color();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Desen.DrawImage(img,0,0);
            for (int j = 0; j < 309; j++)
            {
                for (int i = 0; i < 295; i++)
                {
                    //img1.SetPixel(j, i, img.GetPixel(i, j));
                    cul = img.GetPixel(i, j);
                    img1.SetPixel(j, i, cul);
                }
            }
            Desen.DrawImage(img1, 300, 0);
        }
    }
}
```

Lucrari de laborator

- Sa se calculeze media aritmetica a elementelor unui tablou unidimensional.
- Sa presupunem ca avem un tablou cu 31 elemente, reprezentand consumurile zilnice de energie. Calculati consumul lunar de energie.
- Avand un tablou unidimensional cu n elemente, sa se construiasca un nou tablou din elementele primului, elemente care sunt mai mari decat un nr a

- Afisati maximul si minimul unui tablou unidimensional
- Se citesc doua tablouri unidimensionale cu componente numere naturale. Fiecare tablou are elementele sortate crescator. Se cere sa se construiasca un al treilea tablou care contine elementele celor doua in ordine crescatoare.
- Aranjati un tablou unidimensional astfel incat numerele pare sa ocupe primele locuri
- Avand doua tablouri unidimensionale a, b , afisati elementele comune, elementele ce apar numai in tabloul a, elementele ce apar numai in tabloul b
- Realizati un tablou bidimensional de dimensiune $n \times n$ realizat din numere pozitive
- Afisati minimul si maximul unui tablou bidimensional de dimensiune $n \times n$
- Afisati elementele de pe diagonala principala unui tablou bidimensional de dimensiune $n \times n$
- Afisati elementele de pe diagonala secundara unui tablou bidimensional de dimensiune $n \times n$

