TABLOURI BIDIMENSIONALE

1. SCOPUL LUCRĂRII

În această lucrare se prezintă aplicaţii cu tablouri bidimensionale (matrici) si aplicaţii cu şiruri de caractere (stringuri).

2. BREVIAR TEORETIC

2.1. Tablouri bidimensionale

Un tablou reprezintă o colecţie de date de acelaşi tip. Tablourile bidimensionale sunt denumite şi matrici. La declararea unui tablou se specifică numărul de elemente ale fiecărei dimensiuni, incluzând fiecare dintre aceste numere între paranteze drepte. Indexul inferior al fiecărei dimensiuni este 0.

Sintaxa pentru declararea unei matrici este următoarea:

tipul_datelor nume_matrice[nr_linii][nr_coloane];

Exemplu: a : o matrice de numere întregi , ce are 10 linii a câte 5 componente pe linie :

int a[10][5];

În general la tablourile multidimensionale şi în particular la matrici, indexul cel mai din dreapta variază cel mai rapid, adică pentru exemplul tabloului a, elementele componente sunt stocate în memorie, în următoarea ordine:

a[0][0], a[0][1], ..., a[0][4],
a[1][0], a[1][1], ...

Un tablou bidimensional poate fi iniţializat încă din faza de declarare, ca în exemplul următor:

```
int mat[3][2]={ 0, 1,
                3, 3,
                4, 8};
```

3. DESFĂŞURAREA LUCRĂRII

Se vor edita şi apoi executa programele descrise în continuare.

Programul nr. 1

Să se calculeze şi să se afişeze suma a două matrice de câte 3 linii şi 4 coloane fiecare.

Vom iniţializa prin atribuire directă, elementele celor două matrici.

Sursa programului:

```
#include <stdio.h>
```

```
#include <conio.h>
#define N 3 //numarul de linii ale matricilor
#define M 4 //numarul de coloane ale matricilor
void main(void)
{
 int A[N][M]={ 0, 1, 2, 5,
               2, 2, 2, 2,
               3, 3, 3, 3 };
 int B[N][M]={ 1, 5, 7, 9,
               1, 3, 7, 0,
               4, 4, 6, 2 };
 int C[N][M]; //suma matricilor A si B
 int i, j;
 for(i=0;i<N;i++)
        for(j=0;j<M;j++)
                C[i][j]=A[i][j]+B[i][j];
 //afisarea matricii C :
 clrscr();
 for(i=0;i<N;i++){
        printf("\n");
        for(j=0;j<M;j++)
                printf("%d  ",C[i][j]);
 }
 getch();
}
```

Programul nr. 2

Să se substituie într-o matrice pătrată toate elementele aflate sub diagonala principala cu valoarea 0. Dimensiunea matricii se citeşte de la tastatură.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a[50][50]; //o dimensiune acoperitoare
 int n;//dimensiunea  matricii
 int i,j;
 clrscr();
 printf("Dimensiunea matricii n=");
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
        for(j=0;j<n;j++){
          printf("a[%d][%d]=",i,j);
          scanf("%d",&a[i][j]);}
//Afisarea matricii a, inainte de substitutie:
printf("Inainte de substitutie:\n");
for(i=0;i<n;i++){
         for(j=0;j<n;j++)
        printf("%d  ",a[i][j]);
         printf("\n");}
//Substitutia cu 0:
for(i=1;i<n;i++)
        for(j=0;j<i;j++)
          a[i][j]=0;
//Afisarea matricii a, dupa substitutie:
printf("Dupa substitutie:\n");
for(i=0;i<n;i++){
         for(j=0;j<n;j++)
        printf("%d  ",a[i][j]);
         printf("\n");}
getch();
}
```

Programul nr. 3

Pentru o matrice pătrată A de numere întregi, să se calculeze şi afişeze care este numărul (indexul) coloanei ce are suma elementelor (ale coloanei respective) maximă faţă de sumele elementelor din celelalte coloane ale matricii.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 3  //numarul de linii (sau coloane) din matricea A
void main(void)
{
 int A[N][N]={ 0, 1, 2,
               2, 2, 2,
               3, 3, 3 };
 //(Pentru acest exemplu, vom obtine ca rezultat coloana 2, pentru ca
 //ea are suma maxima 2+2+3=7)
```

```
 int i,j;
int sumaCrt; //suma elementelor coloanei curente
 int sumaMax; // maximul cautat
 int indexSumaMax; //indexul cautat
 clrscr();
 //initializam  sumaMax cu suma elementelor primei coloane:
 sumaMax=0;
 for(i=0;i<N;i++)
        sumaMax=sumaMax+A[i][0];
 //initializare indexSumaMax:
 indexSumaMax=0;
 //Calculam sumele elementelor din restul coloanelor:
 for(j=1;j<N;j++){
        //calculul sumei elementelor coloanei j:
        sumaCrt=0;
        for(i=0;i<N;i++)
               sumaCrt=sumaCrt+A[i][j];
        //este mai mare ca sumaMax ?
        if(sumaCrt > sumaMax){
               sumaMax=sumaCrt;
               indexSumaMax=j; }
 } //end for j
 printf("\nColoana cautata = %d ", indexSumaMax);
 printf("\nAre suma= %d .", sumaMax);
}
```

Programul nr.4

Să se creeze o matrice pătrata de dimensiune N, iniţializata cu numere aleatoare, ce conţine toate numerele din mulţimea: {0,1,2,...,N*N-1}, fiecare element al mulţimii apărând o singura data.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define N 3 //dimensiunea matricii patrate
int  void main()
{
 int A[N][N];
 int i,j;
 int nr;
```

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

```
 int aMaiFostGenerat[N*N];//aMaiFostGenerat[i] este 1, daca numarul i a
 // mai fost generat anterior, deci este deja scris in matrice.
 //Initializare vector aMaiFostGenerat[] cu 0 (fals), deoarece la inceput,
 // nici un numar nu a fost folosit in matrice.
 for(i=0;i<N*N;i++)
   aMaiFostGenerat[i]=0;
 randomize();          srand(time(NULL));
 for(i=0;i<N;i++)
  for(j=0;j<N;j++){
   for(;;){
    nr=random(N*N);//nr este cuprins intre 0 si N*N-1      nr=rand()%(N*N);
    if(aMaiFostGenerat[nr]==0){
        aMaiFostGenerat[nr]=1;
        break;}
    }
   //scriem numarul generat in matrice:
   A[i][j]=nr;
  }
 //afisare matrice:
 clrscr();
 for(i=0;i<N;i++){
  for(j=0;j<N;j++)
   printf("%d  ",A[i][j]);          cout<<A[i][j];
  printf("\n");                     cout<<"\n";
 }
 getch();
}
```

Programul nr. 5

Să se copieze toate elementele unei matrici într-un vector în următoarea ordine: întâi prima linie, apoi a doua linie, etc.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 3 //numarul de linii
#define M 2 //numarul de coloane
void main()
{
```

```
 int A[N][M]={1,2,
              0,1,
              4,3};
  int V[N*M];
  int i,j;
  clrscr();
  for(i=0;i<N;i++)
    for(j=0;j<M;j++)
      V[i*M+j]=A[i][j];
  //afisare vector:
  for(i=0;i<N*M;i++)
    printf("%d  ",V[i]);
  getch();
}
```

Programul nr. 6

Se citesc de la tastatură dimensiunile unei matrici de numere întregi. Se citeşte apoi matricea. Să se afişeze dacă sunt diferite toate elementele matricii între ele sau nu.

Sursa programului:

```
#include <stdio.h>
void main()
{
 int a[20][20];
 int nL;//nr. efectiv de linii din matrice
 int nC;//nr. coloane
 int i,j;
 printf("nr. linii = "); scanf("%d",&nL);
 printf("nr. coloane = "); scanf("%d",&nC);
 for(i=0;i<nL;i++)
  for(j=0;j<nC;j++){
   printf("a[%d][%d]=",i,j);
   scanf("%d",&a[i][j]);}
 //Parcurgem toate elementele matricii, ca si cum ar fi dispuse
 // intr-un vector, de la primul, pana la penultimul:
 for(i=0;i<nL*nC-1;i++)
  //compar elementul curent, cu toate elementele dupa el:
  //(elementul curent este in linia i/nC si in coloana i%nC)
  for(j=i+1;j<nL*nC;j++)
    if(a[i/nC][i%nC]==a[j/nC][j%nC]){
```

```
      printf("Nu sunt toate diferite.");
      return;}
 printf("Sunt toate diferite.");
}
```

Programul nr.7

Scrieţi un program care verifică dacă o matrice este pătrat magic.
Se va verifica mai întâi dacă în matrice sunt prezente toate valorile din mulţimea 1,2,...,N*N.

Sursa programului:

```
/* O matrice patrata de dimensiune N, ce contine toate numerele 1,2,..,N*N,
 este patrat magic daca suma elementelor din fiecare linie, din fiecare coloana, de pe
 diagonala principala si de pe diagonala secundara, are aceeasi valoare.
 Astfel, urmatoarea matrice, este patrat magic:
    16  3  2 13
     5 10 11  8
     9  6  7 12
     4 15 14  1
*/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define N 4 //dimensiunea matricii
#define NxN N*N //numarul de elemente din matrice
void main()
{
 int A[N][N]={16, 3, 2,13,
              5,10,11, 8,
              9, 6, 7,12,
              4,15,14, 1};
 int i,j;
 int suntPrezente[NxN];
 int S;//valoarea comuna, daca este patrat magic
 int sCrt;//suma curenta calculata
 clrscr();
 //Verificam daca matricea contine toate numerele din multimea
//{1,2,...,N*N}:
```

```
//Fiecare numar valid prezent, il inregistram in vectorul
//suntPrezente.
for(i=0;i<NxN;i++)
  suntPrezente[i]=0;
for(i=0;i<N;i++)
 for(j=0;j<N;j++)
   if((A[i][j]<1)||(A[i][j]>NxN)){
      printf("Nu are toate elementele in gama 1,..,N*N .");
      getch();
      exit(1);}
   else suntPrezente[A[i][j]-1]=1;
for(i=0;i<NxN;i++)
  if(suntPrezente[i]==0){
      printf("Nu are toate elementele diferite, in gama 1,..,N*N .");
      getch();
      exit(1);}
//Calculam suma elem. de pe diagonala principala:
S=0;
for(i=0;i<N;i++)
  S=S+A[i][i];
//Comparam valoarea S cu suma sCrt a fiecarei linii:
for(i=0;i<N;i++){
  //calculam suma de pe linia i:
  sCrt=0;
  for(j=0;j<N;j++)
    sCrt=sCrt+A[i][j];
  if(sCrt!=S){
     printf("Nu este patrat magic.");
     getch();
     exit(1);}
 }
 //Comparam valoarea S cu suma sCrt a fiecarei coloane:
for(i=0;i<N;i++){
  //calculam suma de pe coloana i:
  sCrt=0;
  for(j=0;j<N;j++)
    sCrt=sCrt+A[j][i];
  if(sCrt!=S){
     printf("Nu este patrat magic.");
     getch();
```

```
    exit(1);}
 }
//Comparam valoarea S cu suma sCrt de pe diagonala secundara:
sCrt=0;
for(i=0;i<N;i++)
 sCrt=sCrt+A[i][N-i-1];
if(sCrt!=S)printf("Nu este patrat magic.");
else printf("Este patrat magic.");
getch();
}
```

Programul nr.8

Implementaţi următorul algoritm prin care se construieşte un pătrat magic, de dimensiune N. Acesta este valabil numai pentru N - impar.
Plasaţi valoarea 1 in mijlocul rândului de jos.
Apoi, pentru fiecare număr Nr de la 2 la N*N , se repetă:
dacă numărul anterior (Nr-1) a fost plasat in linia i si coloana j, numărul curent Nr se va plasa in linia i+1 (o linie mai jos) si în coloana j+1 (o coloana mai la dreapta). Daca linia trece peste ultima: se va trece in linia 0. La fel si daca se trece peste ultima coloana: se trece în coloana 0. Daca poziţia calculată este deja ocupată, numărul curent se va scrie în aceeaşi coloană, dar cu o linie mai sus.
Astfel pentru N=3, se obţine următorul pătrat magic:

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

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 3 //dimensiunea matricii
void main()
{
  int a[N][N];
  int i,j,linCrt,colCrt,lin,col;
  int nr;//numarul curent ce se plaseaza in matrice
  clrscr();
  //Umplem matrice cu 0: nu e ocupata:
  for(i=0;i<N;i++)
    for(j=0;j<N;j++)
      a[i][j]=0;
```

```
  //punem pe 1 in matrice, in linia N-1 (ultima linie)
  // si coloana N/2 (cea din mijloc):
  linCrt=N-1;
  colCrt=N/2;//n este impar
  a[linCrt][colCrt]=1;
  for(nr=2;nr<=N*N;nr++)
  {
   //linia curenta si coloana curenta sunt pentru coltul dreapta jos?
   if((linCrt==N-1)&&(colCrt==N-1))linCrt=N-2;
   else{
         //salvam linia si coloana curenta, in caz ca gasim ocupat:
         lin=linCrt;
         col=colCrt;
         //mergem in  dreapta jos:
         linCrt=linCrt+1;
         colCrt=colCrt+1;
         //a iesit din margini ?
         if (linCrt==N)linCrt=0;
         if (colCrt==N)colCrt=0;
         //e ocupat?
         if(a[linCrt][colCrt]!=0){
           linCrt=lin-1;
           colCrt=col;}
    }//else
    a[linCrt][colCrt]=nr;
   }//for
   //afisarea matricii rezultate:
   for(i=0;i<N;i++){
     for(j=0;j<N;j++)
      printf("%d ",a[i][j]);
     printf("\n");
   }
  getch();
}
```

4. PROBLEME PROPUSE

1. Se citesc 16 numere întregi de la tastatură . Aceste numere trebuie memorate într-o matrice pătrată de 4 linii şi 4 coloane. Să se calculeze şi afişeze maximul din matrice cât şi linia şi coloana în care

apare acest maxim. (În cazul în care sunt mai multe valori egale cu maximul, se va afişa linia şi coloana primei apariţii).

2. Să se comute două coloane ale unei matrici între ele. Matricea şi numerele coloanelor care se comută, se vor citi de la tastatură.

3. Să se verifice dacă o matrice pătrată NxN este simetrică (faţă de diagonala principală) .

4. Se citesc mai multe numere întregi, într-o matrice. Se citeşte de asemenea numărul unei linii. Să se copieze linia respectivă într-un vector.

5. Să se permute două linii ale unei matrici între ele. Numerele celor doua linii, ca şi matricea, se vor citi de la tastatură.

8. Să se scrie un program ce verifică dacă o matrice pătrată NxN are toate elementele de sub diagonala principală nule.