**LUCRAREA DE LABORATOR Nr. 4**

*Tema:* **Tehnicile prelucrării tablourilor** **bidimensionale**

*Scopul lucrării:* însuşirea tehnicilor de algoritmizare şi programare cu subprograme în prelucrarea structurilor complexe în TP şi C.

*Obiectivele temei*

1. Aprofundarea cunoştinţelor în limbajul şi mediul TurboC şi perfecţionarea tehnicii de programare în prelucrarea structurilor complexe, analizând soluţiile stereotipe şi cele eficiente de introducere, afişare şi rearanjări ale tablourilor bidimensionale etc.
2. Însuşirea tehnicilor eficiente de parcurgere, căutare, schimbare şi ordonare a matricilor şi calculul conform cunoştinţelor obţinute din matematică şi în baza ciclurilor încorporate într-un program complex.

*Subiectele temei şi ordinea executării*

1. Studierea principiilor prelucrării (descrierii, declarării, formării, etc.) tablourilor bidimensionale .
2. Însuşirea tehnicilor moderne de elaborare a programelor complexe în C în baza problemelor din anexă.
3. Elaborarea algoritmilor, programelor şi testelor de depanare a programului prin diverse moduri în mediul integrat C.

*Conţinutul raportului (vezi lucr. de laborator nr.1-3) şi s*uplimentar: Analiza erorilor admise pe parcursul efectuării lucrării şi eficienţa algoritmilor elaboraţi.

***Consideraţiile teoretice şi exemple***

I. Tablouri bidimensionale

**1.1. Tablouri bidimensionale în TP**. Tablourile care grupează date de acelaşi tip, a căror referire se va face prin poziţia lor în şir, respectiv prin linie şi coloană vor fi tablouri bidimensionale şi vor fi o reprezentare similară a matricelor. Atunci dacă în sintaxa tablourilor unidimensionale (vezi lucr. laborator nr. 3) adăugăm încă o pereche de *index\_inferior..index\_superior* separate prin virgulă vom avea un tablou bidimensional, adică matricea, iar dacă înşirăm mai multe perechi de *index\_inferior..index\_superior* separate prin virgulă atunci vom avea un tablou cu mai multe dimensiuni.

Sintaxa: *Type*

*tip\_tablou=Array[index\_inferior..index\_superior, index\_inferior..index\_superior] of tip\_element;*

urmînd ca variabila tablou bidimensional s-o declarăm în mod obişnuit:

*Var nume\_tablou : tip\_tablou;*

Exemple:  *Type matrice=array[1..50,1..50] of integer;*  ar însemna matrice patratică cu 50 linii şi coloane. Elementele matricei sunt numere întregi, numerotate pe linie şi coloane de la 1 la 50. Fie acum o variabilă de acest tip:

*Var a : matrice;* . Atunci al cincilea element de linie şi al treilea pe coloană din tabloul bidimensional *a* se referă prin a[5,3], deci componenta i şi j se va specifica prin a[i,j].

**1.2. Tablouri bidimensionale în C.** Tabloul bidimensional reprezintă o structură formată din rînduri şi coloane. Fiecare element al unui tablou bidimensional are 2 coordonate: numarul rîndului şi numărul coloanei. Deci matricile sunt tablouri bidimensionale şi pot fi asemuite cu o secvenţa de mai multe şiruri. În C dacă un element al unui şir se identifică cu elem[i] unde i reprezintă poziţia acelui element în cadrul şirului, la matrici un element se identifica prin *elem[i][j],* unde i reprezintă poziţia pe coloană (verticală) a acelui element, iar j reprezintă poziţia pe linie (orizontală) a acelui element. Dacă facem analogia cu şirurile, i reprezintă numărul şirului în care se află elementul, iar j reprezintă poziţia elementului în cadrul şirului i.



O matrice va avea ca şi număr de elemente produsul dintre numărul de coloane şi numărul de linii. Dacă notăm aceste valori cu m (vertical) respectiv cu n (orizontal) atunci o matrice Am·n m · n elemente. În exemplul dat mai sus, m = 3 şi n = 4. Ca şi în cazul şirurilor cei doi indici, **i şi j**  vor lua valori între **0 şi m-1 respectiv n-1.**

Dacă trebuie *declarat un* tablou *de valori(tablou)* de un anumit tip se va indica între paranteze drepte numărul de elemente din tablou. Exemple: *int* a, b[100], c=0; *double* p[30], q[20][30];

Pentru tabloul b compilatorul va rezerva 100 de celule consecutive pentru 100 de valori întregi. Indicele folosit la selectarea unei valori oarecare poate varia între 0 si 99.

Variabila q corespunde unei matrici bidimensionale având 20 de linii si 30 de coloane. Indicii se scriu separat, între paranteze drepte si pot varia de la 0 la 19 respectiv de la 0 la 29. Se poate omite indicele al doilea, q[i] desemnând linia i.

Limbajul C permite Tablouri de orice tip, inclusiv Tablouri de Tablouri. Putem obtine Tablouri de dimensiune 2, 3, ... . Exemple:

int a[100]; <- tablou de dimensiune 1

int b[2][7]; <- tablou de dimensiune 2

int c[5][3][2]; <- tablou de dimensiune 3

Pornind de la adresa de baza, toate elementele tabloului sunt memorate contiguu in memorie. Prin definiţie un tablou bidimensional este de fapt un tablou unidimensional ale carei elemente sunt fiecare in parte câte un tablou. Prin urmare, indicii se scriu astfel a[i][j] in loc de a[i, j] ca in majoritatea limbajelor. In plus un tablou bidimensional poate fi tratat in mai multe moduri decât in alte limbaje. Elementele sunt memorate pe linii, ceea ce inseamna ca indicele din dreapta variaza primul in aşa fel incit elementele sunt accesate in ordinea memoriei.

Vectori 2-dimensionali

Presupunem ca avem un vector 2-dimensional cu elemente intregi. int a[3][5];

Incepand cu adresa de baza, compilatorul va aloca spatiu contiguu pentru 15 intregi. Atunci putem gândi acest vector ca o matrice, astfel:

*col1 col2 col3 col4 col5*

*lin1* a[0][0] a[0][1] a[0][2] a[0][3] a[0][4]

*lin2* a[1][0] a[1][1] a[1][2] a[1][3] a[1][4]

*lin3* a[2][0] a[2][1] a[2][2] a[2][3] a[2][4]

Pentru a[i][j] avem expresiile, utilizând pointeri, de exemplu, echivalente:

*\*(a[i] + j)*

*(\*(a + i))[j]*

*\*((\*(a + i)) + j)*

*\*(&a[0][0] + 5\*i + j)*

Putem gândi "a[i]" ca a "i"-a coloana a lui "a" (numarand de la 0), si "a[i][j]" ca elementul din linia "i", coloana "j" a tabloului (numarand de la 0). Numele tabloului ("a") este tot una cu "&a[0]"; acesta este un pointer catre un tablo de 5 intregi. Adresa de baza este "&a[0][0]", si nu "a". Ultimul exemplu de mai sus reflecta functia de corespondenta in memorie dintre valoarea pointerului si indicele tabloului.

Când un vector multidimensional este un parametru formal in definitia unei functii, toate dimensiunile, exceptand prima trebuie specificate.

Exemplu: Presupunem ca sunt date elementele vectorului "a". Functia de mai jos se poate folosi pentru suma elementelor unui tablou. Atentie ! Trebuie specificat numaidecât numarul de coloane.

**int suma(int a[][5])**

**{**

**int i, j, suma = 0;**

**for (i = 0; i < 3; ++i)**

**for (j = 0; j < 5; ++j)**

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

**return suma;**

**}**

In antetul functiei, urmatoarele declaratii sunt echivalente:

int a[][5] int (\*a)[5] int a[3][5]

Constanta 3 actioneaza ca o reminiscenta a omului, dar compilatorul nu tine cont de ea.

Nou venitii in C sunt uneori confuzi in legatura cu deosebirea dintre un tablou bidimensional si un tablou de pointeri cum ar fi "a" din exemplul de mai sus. Fiind date declaratiile

int a[10][10];

int \*b[10];

utilizarile lui "a" si "b" pot fi similare, in sensul ca a[5][5] si b[5][5] sunt ambele referinte legale ale aceluiasi "int".

Avantaje pentru utilizarea vectorilor (dezavantaje pentru pointeri):

- "a" este un tablou in toata regula: toate cele 100 celule de memorie trebuie alocate, iar pentru gasirea fiecarui element

se face calculul obisnuit al indicelui;

- pentru "b", oricum prin declararea sa se aloca 10 pointeri; fiecare trebuie facut sa pointeze un tablou de intregi.

Presupunind ca fiecare pointeaza cate 10 elemente din tablou, atunci vom obtine 100 celule de memorie rezervate, plus cele 10 celule pentru pointeri. Astfel tabloul de pointeri utilizeaza sensibil mai mult spatiu si poate cere un procedeu explicit de initializare.

Avantaje pentru utilizarea pointerilor (dezavantaje pentru vectori):

- accesarea unui element se face indirect prin intermediul unui pointer, in loc sa se faca prin inmultire si adunare;

* liniile tabloului pot fi de lungimi diferite. Aceasta inseamna ca nu orice element al lui b este constrins sa pointeze pe un vector de 10 elemente, unii pot pointa pe cate 2 elemente, altii pe cate 20 si altii pe niciunul.
* Un identificator de tablou este, în acelaşi timp, un pointer a cărui valoare este adresa primului element al tabloului. Prin această regulă, tablourile sunt identificate cu adresele primelor lor elemente. De exemplu, identificatorul a de mai sus (definit ca int a[2][5]) este de tip pointer la un tablou cu cinci elemente întregi, adică int (\*)[5], iar a[0] şi a[1] sunt adrese de întregi, adică int\*. Mai exact, expresia a[0] este adresa primei linii din matrice (a primului tablou de cinci elemente) şi este echivalentă cu \*(a+0), iar expresia a[1] este adresa celei de-a doua linii din matrice (a celui de-al doilea tablou de cinci elemente), adică \*(a+1). În final, deducem că a[1][2] este echivalent cu \*(\*(a+1)+2), ceea ce ilustrează echivalenţa operatorului de indexare şi a celui de indirectare.

În privinţa echivalenţei identificatorilor de tablouri şi a pointerilor, nu mai putem fi atît de categorici. Să pornim de la urmatoarele două definiţii:

int a[ 2 ][ 5 ];

int \*b[ 2 ] = {

a[ 0 ] // adica b[ 0 ] = &a[ 0 ][ 0 ]

a[ 1 ] // adica b[ 1 ] = &a[ 1 ][ 0 ]

};

unde a este un tablou de 2 \* 5 elemente întregi, iar b este un tablou de două adrese de întregi.

Evaluînd expresia b[1][2], obţinem \*(\*(b+1)+2), adică elementul a[1][2], element adresat şi prin expresia echivalentă \*(\*(a+1)+2). Se observă că valoarea pointerului \*(b+1) este memorată în al doilea element din b (de adresa b+1), în timp ce valoarea \*(a+1), tot de tip pointer la int, nu este memorată, fiind substituită direct cu adresa celei de-a doua linii din a.

Tratarea diferită a expresiilor echivalente \*(b+1) şi \*(a+1) se datorează faptului că identificatorii de tablouri nu sunt de tip pointer, ci de tip pointer constant. Valoarea lor nu poate fi modificată, deoarece este o constantă rezultată în urma compilării programului

*Vectori 3-dimensionali*

Vectorii de dimensiune mai mare decat 3 lucreaza intr-un mod similar. Daca avem declaratia

int a[7][9][2];

atunci compilatorul va aloca spatiu pentru 7\*9\*2 intregi. Adresa de baza a tabloului este "&a[0][0][0]", iar functia de corespondenta in memorie este specificata de

a[i][j][k] care este echivalent cu \*(&a[0][0][0] + 9\*2\*i + 2\*j + k)

**Initializarea tablourilor**

Exista mai multe moduri de a initializa un vector multidimensional.

Exemplu: *matrix[4][7 ] -* se accesează elementul de pe rîndul 4 şi coloana 7 a tabloului matrix.

*y[0][0]* se accesează primul element al tabloului, adică rîndul 0, coloana 0.

În cazul cînd tabloul este de tip simplu, atribuirea valorii unui element al tabloului se face ca şi în cazul atribuirii valorii unei variabile simple. Exemple:

*x[0]=7.125;*

*vector[19]+=1;*

*matrix[1][1]=5.5;*

*fraza[3]=’b’;*

În cazurile cînd tabloul este de tip structurat, atribuirea valorii şi accesul laun element al tabloului se face conform regulilor de atribuire şi accesare pentru variabile structurate. Un element al tabloului poate apărea în orice expresie, unde e admisibilăprezenţa unei variabile de tip compatibil cu tipul valorii elementului.

Să examinăm iniţializarea tabloului bidimensional:

Int a[3][3]=

{ {1,4,2},

{7,5,3},

{8,6,9} };

Iniţializarea tabloului bidimensional se efectuează pe rînduri. Elementele acestui tablou au următoarele valori:

a[0][0]=1; a[0][1]=4; a[0][2]=2;

a[1][0]=7; a[1][1]=5; a[1][2]=3;

a[2][0]=8; a[2][1]=6; a[2][2]=9;

La iniţializarea tabloului dat fiecare rînd se include în acolade . Dacă în rîndurile indicate de noi nu vor ajunge elemente pentru completarea rîndurilor, în acest caz în locul elementelor pentru care n-au ajuns valori vor apărea zerouri.

Exemplu: Urmatoarele declaratii sunt echivalente:

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

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

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

Indexarea se face dupa linii. Daca nu sunt suficiente elemente care sa initializeze vectorul, atunci restul elementelor sunt initializate cu 0. Daca prima componenta lipseste, atunci compilatorul extrage lungimea din numarul de perechi de acolade interioare.

Exemplu: Consideram initializarea:

int a[2][2][3] = {

{{1, 1, 0}, {2, 0, 0}},

{{3, 0, 0}, {4, 4, 0}}

};

O initializare echivalenta poate fi data si astfel:

**int a[][2][3] = {{{1, 1}, {2}}, {{3}, {4, 4}}};**

De obicei, daca un tablou declarat "auto" nu este explicit initializat, atunci elementele tabloului vor contine valori "garbage".

Tablourile "static" si "external" sunt initializate implicit cu 0. Iata un mod simplu de a initializa toate valorile unui vector cu 0: **int a[2][2][3] = {0};**

**Identificarea elementelor specifice unei matrici pătratice** În cazul matricilor pătratice se identifică următoarele elemente specifice: diagonala principală (**DP**), diagonala secundară (**DS**), jumătatea superioară (**JS**) şi jumătatea inferioară (**JI**). Diagonalele corespund (geometric) diagonalelor unui pătrat.

Diagonala principală cuprinde elementele din colţul stânga sus până în colţul dreapta jos, adică mulţimea:

**DP={aij| i=j, i=0,…..n-1}={a00, a11, a22, ……………….,an-1n-1}** unde **n** este dimensiunea reală a matricii.

Diagonala secundară cuprinde elementele din colţul dreapta sus până în colţul stânga jos, adică mulţimea:

**DS={aij|i=0,1,…n-1 , j=n-1,n-2,…0, i+j=n-1}={a0n-1, a1n-2,……..,an-10}**

Jumătatea superioară cuprinde elementele de deasupra diagonalei principale (fără cele de pe **DP**), adică mulţimea:

**JS={aij | i=0,1,………n-2 , j=i+1,…………………,n-1}**

Jumătatea inferioară cuprinde elementele de sub diagonala principală (fără elementele de pe **DP**), adică mulţimea:

**JI={aij | i=1,2,……….n-1 , j=0,1,………i-1}**

Un element **a[i][j]** din **JS** are drept simetric elementul **a[j][i]** din **JI**. Astfel, transpusa unei matrici pătratice (matricea care are liniile drept coloane şi reciproc) se poate genera uşor interschimbând toate elementele din **JS** cu simetricele lor din **JI** aşa cum se vede în secvenţa de mai jos:

**for(i=0; i<n-1; i++)**

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

**{ aux=a[i][j];**

**a[i][j]=a[j][i];**

**a[j][i]=aux; }**

**Întrebări – răspuns** pe baza teoriei prezentate:

1. Ce reprezinta un identificator de tablou?

Un identificator de tablou, este un pointer a carui valoare este adresa primului element din tablou.

1. Care este condiţia ca o matrice să fie triunghiulară?

O matrice este triunghiulară, atunci cînd doar elementele situate de-o parte a diagonalei principale sunt efectiv utilizate.

1. Ce se întîmplă atunci cînd liniile unei matrici nu au acelaşi număr de elemente?

Dacă liniile unei matrici nu au acelaşi număr de elemente, atunci:

* + În tablourile predefinite, fiecare linie este de lungime maximă.
  + În tablourile bazate pe clasă tablou<T>, fiecare linie poate fi dimensionată corespunzător numărului efectiv de elemente.

Fie programul următor :

**void main()**

**{ int i,j,m,n,p,a[10][10],b[6]; m=2; n=3; p=6; i=0;**

**while(i<p) b[i++]=i;**

**for(i=0;i<m;i++)**

**for(j=0;j<n;j++) a[i][j]=b[3\*i+j]; }** În urma execuţiei sunt posibile următoarele situaţii:

a) programul nu funcţionează din cauză că declaraţia matricei este eronată

b) valorile vectorului **b** sunt 0,1,2,3,4,5

c) valorile vectorului **b** sunt 1,2,3,4,5,6

d) **a[1][0]** are valoarea 3

e) **a[0][2]** are valoarea 2

7. Se consideră o matrice **a** cu **n** linii\***n** coloane şi un vector **v** cu **n** elemente. Precizaţi care vor fi elementele vectorului **v**, după execuţia secvenţei următoare:

**int nr, n, i, j, x, b[20]; int a[3][3]={{7,1,7},{-7,7,0}, {2,4,11}}; n=3; x=7;**

**for(i=0;i<n;i++) { nr=0;**

**for(j=0;j<n;j++) if(a[i][j]==x) nr++; b[i]=nr; }**

a) nedefinite b) v=(0,0,0)

c) v=(1,2,3) d) v=(2,0,1)

e) v=(2,1,0)

8. Se consideră secvenţa de program următoare, în care **a** este o matrice cu **n** linii\***n** coloane şi elemente numere întregi, iar **x** este o variabilă de tip întreg.

**x=1;**

**for(i=1;i<=n-1;i++)**

**for(j=0;j<=i-1;j++) if(a[i][j]!=0) x=0;**

În urma execuţiei secvenţei, valoarea variabilei **x** va fi 1 dacă:

a) deasupra diagonalei principale există cel puţin un element egal cu 0

b) toate elementele de deasupra diagonalei principale sunt 0

c) toate elementele de sub diagonala principală sunt diferite de 0

d) toate elementele de sub diagonala principală sunt 0

e) sub diagonala principală există cel puţin un element diferit de 0

Se consideră matricea pătratică **A(mxm**) . Fie secvenţa de program :

**x=a[0][m-1];**

**for(i=0;i<m;i++) if x<a[i][m-i] x=a[i][m-i];**

Variabila **x** calculată exprimă :

a) valoarea maximă de pe diagonala principală

b) valoarea maximă de pe diagonala secundară

c) valarea maximă din întreaga matrice

d) altă valoare decat cele indicate

26. Fie matricea pătratică **A(mxm)** şi secvenţa de program :

**x=a[0][0];**

**for(i=0;i<m;i++) { if(x<a[i][i]) x=a[i][i]; if(x<a[i][m-i-1]) x=a[i][m-i-1]; }**

Variabila **x** calculată reflectă :

a) valoarea cea mai mare de pe diagonale

b) valoarea cea mai mare de pe diagonala principală

c) valoarea cea mai mare de pe diagonala secundară

d) altă valoare decat cele indicate

27. Fie matricea **A(mxm)** şi secvenţa de program:

**s=1;**

**for(i=0;i<m;i++) for(j=0;j<i;j++) s\*=a[i][j];**

Valoarea calculată **s** reflectă :

a) produsul valorilor de sub diagonala principală

b) produsul valorilor de pe diagonala secundară

c) produsul valorilor din întreaga matrice

d) altă valoare decât cele indicate

28. Fie o matrice **A(nxn)** citită de la tastatură . Atunci secvenţa de cod :

**i=0; m=a[0][0];**

**while(i<n) { for(j=0;j<n;j++) if(m<a[i][j]) m=a[i][j]; i++; }**

a) calculează elementul maxim **m** dintre elementele matricii

b) calculează elementul minim **m** dintre elementele matricii

c) se ciclează la infinit

d) numără elementele matricii care sunt mai mari ca **m**

29. Fie secvenţa de cod de mai jos. Dacă considerăm ca date de intrare tabloul **V1** de dimensiune **n**, atunci tabloul **V0** va conţine :

**for(i=0;i<n;i++) v0[n-i-1]=v1[i];**

a) elementele lui **V1** în ordine inversă

b) o parte din elementele lui **V1**

c) numai elem. impare ale lui **V1**

d) algoritmul depune elementele lui **V1** în ordine inversă numai dacă **n** este impar

30. Fie următoarea secvenţă de cod care primeşte o matrice **M** pătratică de dimensiune **n** la intrare :

**p=1;**

**for(i=0;i<n;i++) p=p\*m[i][i]-2;** atunci aceasta realizează :

a) produsul elementelor din matrice

b) produsul elementelor de pe diagonala principală a matricii **m**

c) o prelucrare a elementelor de pe diagonala principală a matricii **m**

d) se ciclează la infinit

31. Care este valoarea elementului **tab[2][3]** după execuţia secvenţei de cod :

**int i,j; int ctr=0; int tab[4][4];**

**for(i=0;i<4;i++)**

**for(j=0;j<4;j++) { tab[i][j]:=ctr; ++ctr; }**

a) 7 b) 9 c) 11 d) 14

***Exemple model:***

***Scopul lucrarii :*** Prelucrarea matricei pe coloane: calculul sumei valorilor elementelor pe coloane:

#include <stdio.h>

#define ROW 2

#define COL 3

void main() {

int a[ROW][COL], b[ROW]; int i,j;

for(i = 0; i < ROW; i++) { printf("\n %d\n", i + 1);

for(j = 0; j < COL; i++) scanf("%d",&a[i][j]); }

/\* \*/

for(j = 0; j < COL; i++) { b[j] = 0;

for(i = 0; i < ROW; i++) b[j] = b[j] + a[i][j]; } printf("\n \n");

for(i = 0; i < COL; i++) printf("%d",b[i]); }

***Scopul lucrarii :*** Prelucrarea matricei pe linii: calculul numărul valorilor nule ale elementelor

#include <stdio.h>

#include <conio.h>

#define ROW 2

#define COL 3

void main(){ int a[ROW][COL], b[ROW]; int i, j, n; clrscr();

for (i = 0; i< ROW; i++) {printf("%d\n", i + 1);

for (j = 0; j < COL; i++) scanf("%d",&a[i][j]); }

for (i = 0; i< ROW; i++) { b[i] = 0; n = 0;

* for (j = 0; j < COL; i++) if(a[i][j] == 0) n = n + 1; b[i] = n; }

printf("\n \n");

for (i = 0; i< ROW; i++) printf("%4d",b[i]); }

***Scopul lucrarii :***Să se scrie un program prin care să se calculeze valoarea matricii S = A+B.

#include<stdio.h>

#include<conio.h>

void main()

{ int a[10][10], b[10][10], s[10][10]; int m,n,i,j;

clrscr();

printf("Introduceti dimensiunea matricei\nDati numarul de coloane m= "); scanf("%d",&m);

printf("Dati numarul de linii n= "); scanf("%d",&n);

printf("Introduceti elementele matricei A\n");

for (i=0;i<m;i++)

for(j=0;j<n;j++)

{

printf("Introduceti elementul a[%d][%d]=",i,j);

scanf("%d",&a[i][j]);

}

printf("Introduceti elementele matricei B\n");

for (i=0;i<m;i++)

for(j=0;j<n;j++)

{

printf("Introduceti elementul b[%d][%d]=",i,j);

scanf("%d",&b[i][j]);

}

//calculul sumei

printf("Matricea suma este \n\n");

for (i=0;i<m;i++)

for(j=0;j<n;j++)

s[i][j]=a[i][j]+b[i][j];

//afisare matrici

for (i=0;i<m;i++)

{

for(j=0;j<n;j++)

printf("%4d",s[i][j]);

printf("\n\n");

}

getch();

}

***Scopul lucrarii :*** Este dată matricea A={**a*i,j***}, i=1,..,n; j=1,..,n. Extrageţi vectorul B după schema mişcării reprezentată în Fig.5 (schema mişcării

reprezintă spirala şi punctul iniţial de pornire este **a*i,j***

spre elementul din centrul matricei A Apoi elementele

lui B de rearnjat în creştere.

Fig.5.

**#include<stdio.h>**

**#include<conio.h>**

**void main()**

**{**

**clrscr();**

**int i,n,n1,p,q,aux,r,j,t,b[100],a[100][100];**

**printf("Introdu dimensiunea matricei: n=");**

**scanf("%d",&n);**

**printf("Introdu %d de elementele:\n",n\*n);**

**for(i=0;i<n;i++)**

**for(j=0;j<n;j++)**

**scanf("%d",&a[i][j]);**

**clrscr();**

**printf("Matricea A:\n");**

**for(i=0;i<n;i++)**

**{ printf("\n");**

**for(j=0;j<n;j++)**

**printf("%d ",a[i][j]);**

**}**

**if(n%2!=0) n1=n/2+1;**

**else n1=n/2;**

**p=0;**

**for(i=0;i<n1;i++)**

**{**

**for(j=i;j<n-i;j++)**

**{ b[p]=a[j][i];p++; }**

**for(q=i+1;q<n-i;q++)**

**{ b[p]=a[n-i-1][q];p++; }**

**for(r=n-i-2;r>=i;r--)**

**{ b[p]=a[r][n-i-1];p++; }**

**for(t=n-i-2;t>i;t--)**

**{ b[p]=a[i][t];p++; }**

**}**

**printf("\nVectorul B:\n");**

**for(i=0;i<p;i++) printf("%d ",b[i]);**

**for(i=0;i<p-1;i++)**

**for(j=0;j<p-1;j++)**

**{ if(b[j+1]<b[j])**

**{ aux=b[j];**

**b[j]=b[j+1];**

**b[j+1]=aux;**

**}**

**}**

**printf("\nVectorul B ordonat crescator:\n");**

**for(i=0;i<p;i++)**

**printf("%d ",b[i]);**

**getch();**

**}**

ANEXA Ll. nr.4

**Variantele L.l. “Prelucrarea tablourilor bidimensionale(matricelor)”,**

1. Să se citească de la tastatură o matrice A**m**x**n** .
   1. Să se calculeze suma elementelor strict pozitive.
   2. Să se afiseze poziţia în matrice a elementelor nule. (Ex: Elementul A[2][4] este nul).
2. Să se citească de la tastatură o matrice patratica A**n**x**n** .

Sa se inverseze fiecare element cu negativul lui cu exceptia elementelor de pe diagonale.

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se verifice daca matricea este simetrica iar in cazul in care nu este, sa se schimbe cu 0 toate elementele nesimetrice. (Simetria matricii se verifica fata de diagonala principala)

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se verifice daca matricea este antisimetrica iar in cazul in care nu este, sa se schimbe cu 0 toate elementele nesimetrice. (Antisimetria matricii se verifica fata de diagonala secundara)

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se calculeze suma si produsul elementelor de pe diagonala principala si de sub diagonala secundara

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se calculeze media aritmetica a elementelor pozitive si produsul elementelor de pe diagonala principala si de pe diagonala secundara

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se calculeze suma elementelor negative si produsul elementelor pozitive de sub diagonala principala si de pe diagonala secundara, inclusiv de pe cele doua diagonale.

1. Să se citească de la tastatură o matrice patratica A**n**x**n.**

Sa se calculeze suma elementelor pozitive si produsul elementelor negative de sub diagonala principala si de sub diagonala secundara, inclusiv de pe cele doua diagonale.

1. Se citesc doua matrici patratice A si B. Sa se inverseze elementele de deasupra si de pe diagonala principala intre cele doua matrici. Sa se afiseze matricile inainte si dupa inversarea elementelor.
2. Sa se determine numarul elementelor pozitive dintr-o matrice patratica, care nu se afla pe cele doua diagonale. Sa se gaseasca elementele nule din toata matricea si sa se afiseze pozitiile lor.
3. Se da o matrice patratica. Sa se schimbe toate elementele negative cu -1, iar cele pozitive cu 1. Sa se afiseze noua matrice. Sa se puna apoi toate elementele de pe diagonale in zero si sa se afiseze din nou matricea.
4. Sa se calculeze produsul a doua matrici patratice A si B. Sa se calculeze suma elementelor de deasupra diagonalei principale din matricea produs.
5. Sa se calculeze suma elementelor din triunghiul superior diagonalei principale
6. Sa se calculeze suma elementelor din triunghiul inferior diagonalei principale
7. Sa se calculeze suma elementelor din triunghiul superior diagonalei secundare
8. Sa se calculeze suma elementelor din triunghiul inferior diagonalei secundare
9. Sa se calculeze suma elementelor din triunghiul superior diagonalelor
10. Sa se calculeze suma elementelor din triunghiul inferior diagonalelor
11. Sa se calculeze suma elementelor din triunghiul inferior diagonalei principale si superior diagonalei secundare
12. Sa se calculeze suma elementelor din triunghiul inferior diagonalei secundare si superior diagonalei principale

21. Determinaţi caracteristica coloanei matricii ca suma elementelor negative pe poziţiile pare şi celor pozitive – pe impare. Apoi comparaţi sumele şi în caz că suma celor pozitive este mai mare decît suma celor negative, atunci aranjaţi coloanele matricei date în în ordinea crescătoare, în caz contrar înlocuiţi elementele cu valori pozitive cu valoarea sumei celor pozitive.

22. Pentru matricea dată A={**a*i,j***}, i=1,…,n; j=1,…,n; calculaţi elementele

matricei B={b***i***}, i,j=1,,n; conform schemei următoare:

1

2 3

4

elementul b***i*** va fi egal cu valoarea minimă a laturilor

triunghiurilor obţinute din sectoarele 1…4 în A (Fig.2).

Apoi obţineţi matricea C(C=A\*B\*A). Verificaţi dacă

matricea C este unitară.

Fig.1. Structura matricei A.

23. Pentru matricea dată X={**x*i,j***} de format 2 tablouri bidimensionale după următoarele scheme:

a)**x*1* x*2* .. x*n*** b) **x*n* 0 0**

.............. **x*n-1*** .**x*n* 0**

**x*n-2* x*n-1* x*n* 0 x*n-2* x*n-1* x*n***

**x*n-1* x*n* 0 0 ……………**

**x*n ……* 0 x*1* x*2* x*n***

Apoi de comparat valorile elementelor ambelori tablouri obţinute şi dacă numărul elementelor mai mari din primul tablou este mai mare, atunci în ambele matrici elementele coloanelor impare de schimbat cu locurile celor pare . Fig.3

24. Este dată matricea A={**a*i,j***}; i=1,…,n; j=1,…,n.

Construiţi matricea B={**b*i,j***}, elementele căreia se determină conform următoarei

scheme: în matricea A (Fig.3) prin **a*i,j*** se trasează elementele diagonalelor

paralele cu diagonala principală, iar matricea B cu elementele **b*i,j*** ,

obţinute din **a*i,j***, însă în B aranjate paralel cu diagonala secundară (vezi Fig.4.).

De asemenea determinaţi pentru ambele matrice coordonatele elementelor

minimale şi maximale.

25. Sunt date 2 matrice A={**a*i,j***}, B={**b*i,j***}, delimitate în 4 sectoare (vezi Fig.2.). . Calculaţi pentru ambele matrice următoarele:

valoarea minimă a sectorului 1; -- valoarea maximă a sectorului 2;

produsul elementelor a sectorului 3;. suma elementelor sectorului 4.

26. Pentru matricea dată A={**a*i,j*** }, i=1,..,n; j=1,…,n;

obţineţi vectorul B care reprezintă valoarile minime şi

maxime a elementelor diagonalelor paralele cu

diagonala secundară ( fig.4). Apoi înmulţiţi B la A3 AT.

Afişaţi tablorile obţinute într-o formă clară.

Fig.4.

27. Pentru matricea dată A={**a*i,j***}, i=1,…,n; j=1,…,n; calculaţi elementele matricei B={**b*i,j***}, i=1,n; j=1,n după formula:

**a*i,j* -** **a*k,j***-**a*i,m***/**a*j,i,*** unde k -- numărul liniei care conţine elementul maximal al coloanei m; m -- numărul coloanei care conţine elementul minimal al liniei. Linia k şi coloana m se transmit din A în B neschimbate.

28. Pentru matricea dată WA={**wa*i,j***}, i=1,..,n; j=1,..,n;

extrageţi vectorul WW după schema mişcării

reprezentate în Fig.6, începînd de la elementul **wa*1,n***

spre **wa*2,n*** , apoi paralel cu diagonala principală se

mişcă spre linia întîi, parcurgînd două elemente, şi

dinnou paralel cu diagonala paralelă ajunge la ultima

coloană ş.a.m.d. pînă la sfîrşit la elementul **wa*n,1*** . Fig.6.

29. Elaboraţi algoritmul şi programul pentru prelucrarea unui tablou patratic şi se obţine ca rezultat una din valorile următoare: 0 - dacă tabloul este simetric fata de diagonala principala; 1 - dacă tabloul este superior triunghiular (toate elementele de sub diagonala principală sunt nule) şi 2 - dacă tabloul este inferior triunghiular; 3 - dacă tabloul este unitar şi 4 - în restul cazurilor.

30. Se defineşte că două coloane a matricei date D={**d*i,j***}, i=1,n; j=1,n; sunt asămănătoare dacă coincid seturi de valori ale acestor coloane. De calculat pentru D numărul de coloane cu setul maximal şi numărul de coloane neasămănătoare. *K*

Este dată matricea Q={**q*i,j***}, i=1,n; j=1,n; delimitată în 5

sectoare după schema reprezentată în

fig.7 (pentru ***n*** impar valoarea lui k=n/2+1). Calculaţi:

valoarea minimă a sectorului 1;

valoarea maximă a sectorului 2; -- produsul elementelor al

sectorului 4; suma elementelor a sectorului 3.

31. Din elementele sectorului 5, rombul din centru, creaţi tabloul B={**b*l***}. Fig.7.

Pentru matricea dată Y={**Y*i,j***}, i=1,n; j=1,n; efectuaţi transpunerea faţă de diagonala secundară şi obţineţi matricea B, apoi calculaţi E=Y\***B**3\***Y**T Verificaţi dacă matricea E este singulară sau nu.

32. Pentru matricea dată A={**a*i,j***}, i=1,n; j=1,n; calculaţi complementele algebrice la elementele diagonalei principale şi obţineţi n matrici B de rangul n=1. Matricea B se obţine din A prin excluderea liniei şi coloanei cu numărul i. Afişaţi matriciile obţinute şi elementele lor minime.

33. Sunt date matricile A={**a*i,j***} şi B={**b*i,j***}, i=1,n; j=1,n. Ordonaţi elementele liniilor acestor matrici în ordinea crescătoare. Interclasaţi A şi B şi obţineţi C={**c*i,j***}, i=1,…,n; j=1,…,2n. Elementele liniilor matricii C să fie ordonate în ordinea crescătoare. Apoi calculaţi S=C-A\*B\***B**T

Se consideră date două matrice patratice A si B de ordinul n (n=>20) cu numere întregi. Să se elaboreze algoritmul şi programul pentru efectuarea . calculului elementelor matricei C în baza elementelor matricelor A şi B, definite astfel: deasupra diagonalei principale:

c[i,j]=a[i,j]+b[i,j], dacă cei doi termeni au aceiaşi paritate;

c[i,j]=abs(a[i,j]-b[i,j]), dacă cei doi termeni au parităţi diferite;

34. diagonala principală: semisuma elementelor corespunzatoare din A şi B;

sub diagonala principală:

c[i,j]=a[i,j]+b[i,j], dacă cei doi termeni au parităţi diferite;

c[i,j]=abs(a[i,j]-b[i,j]), dacă cei doi termeni au aceleaşi parităţi.

35. Se consideră date două matrice patratice F si G de ordinul n (n=>20) cu numere întregi. Să se elaboreze algoritmul şi programul pentru efectuarea calculul elementelor matricei H= **F**T\*G-F\***G**T şi de verificat toate matricele prin relaţia de similitudine definită prin urmatoarele trei condiţii:

reflexivitatea elementelor diagonalei principale egale cu 1; simetria lor;

36. tranzitivitatea: orce i,j(1<=i,j<=N) c(i,j)>=max{min(c(i,k),c(k,j)), dacă 1<=k<=n.

37. Se consideră date trei matrice patratice P, R şi S de ordinul n (n=>50) cu numere reale. Să se elaboreze algoritmul şi programul pentru prelucrarea matricelor după următoarele:

se schimbă cu locurile elementele rîndului al doilea cu penultimul;

se schimbă cu locurile elementele coloanei a doua cu penultima;

se schimbă cu locurile elementele coloanelor impare cu valori nule cu respectivele elemente ale rîndurilor impare, adică **p*ji*** =>**p*i,j***.înmulţiţi matricele între ele.

38..Sunt date numerele reale *х1,…,х8*. De obţinut matricele patratice de ordinul 8 A={**a*i,j***} şi B={**b*i,j***}, (n=8) şi de înmulţit între ele de 3 ori, iar din matricea obţinută de determinat valoarea maximă a elementelor situate în sectorul haşurat în Fig.8.

A= ; B= .

Fig.8

39. Se consideră un pătrat cu nxn căsuţe. Fiecare căsuţă conţine un număr între 1 şi 2n. Fiecare număr natural, diferit de 0, apare aleatoriu o singură dată în cadrul pătratului. De găsit calea, începînd din orice linie şi terminînd cu orişice coloană, prin suma minimă a numerelor parcurse.

Traseul de fixat prin înlocuirea numerelor cu 0, ştiind că numărul poate fi schimbat în 0 numai cu cel impar pe poziţile impare. Dacă traseul nu poate fi creat, atunci se caută altă cale. Afişaţi configuraţiile tablourilor iniţială şi finală cu traseul obţinut.

40. Calculati valoarea unui determinant asociat unei matrice patratice. In cazul in care determinantul este nenul, calculati inversa matricei.

41. Calculati inversa unei permutari cu un numar constant de variabile suplimentare.

42. Se considera un labirint cu n la puterea 2 celule. Fiecare celula are cel putin o usa si cel mult trei. Sa se determine drumurile care pot fi parcurse de la celula C11 pana la celula Cnn , astfel incit printr-o usa sa se treaca o singura data.

43. Vom genera siruri de margele formate din mar-gele de diferite dimensuni si de patru culori. Sa se genereze toate sirurile de margele de lungime data in care margelele sunt aranjate in ordine descrescatoare , iar orice doua margele vecine sunt de culori diferite.

## Bibliografie

1. D. Lucanu: Bazele proiectarii programelor si algoritmilor, Universitatea “A.I.Cuza” Iasi, 1996
2. L. Livovschi, H. Georgescu: Sinteza si analiza algoritmilor, Ed. Stiintifica si enciclopedica, 1986
3. O. Catrina, I. Cojocaru, Turbo C++, ed. Teora 1993
4. V. Petrovici, Florin Goicea, Programarea in limbajul C. Eed. Teora 1999
5. Liviu Negrescu, ,,Limbajul C” ,volumul I\_partea I-a si partea II-a. Editura MicroInformatica, Cluj-napoca 2001
6. Б.Керниган, Д.Ритчи. Язык программирования Си. Санкт-Петербург, 2001,,, Brian Kernighan, Dennis Ritchie *” The C Programming Language” este în l. română format electronic*
7. Indrumar de lucrări de laborator: “Informatica”. /A.Popescu, Ş. Marin / Chisinau. UTM,2003.
8. Vlad Caprariu ”Ghid de utilizare Turbo C” Cluj - Napoca 1993.
9. Cristea Valentin. Tehnici de programare. Ed.: Bucur., Teora, 1993. /681.3; T29/
10. Odagescu Ioan, Copos Cristina s.a. Metode si Tehnici de programare./enunturi, solutii, probleme propuse/ Ed.:Bucur.: INTACT, 1994 /681.3; O23/
11. Tudor Bălănescu. Corectudinea agoritmilor.Bucur.:Ed. Tehn.1995