

Programarea calculatoarelor

matrici; vectori și pointeri în contextul matricilor

Multe probleme din teoria grafurilor, rețele neuronale, modelare sau grafică 3D au la bază matrici. În C matricile pot fi văzute ca „vectori de vectori” și reprezintă o extensie naturală a vectorilor. Se poate declara o matrice astfel:

```
int a[m][n];           // o matrice de m linii și n coloane
```

Fiecare dimensiune a matricii se declară între propriile sale paranteze drepte (`[]`). Prima dimensiune (m) declară numărul de linii (*rows* - pe orizontală), iar a doua dimensiune (n) declară numărul de coloane (*columns* - pe verticală).

Atenție: este greșit să se scrie „ $a[m,n]$ ”. Compilatorul va accepta această declarație dar ea va rezulta într-un vector de „ n ” elemente (deoarece „ $[m,n]$ ” înseamnă de fapt aplicarea operatorului secvențial, ceea ce rezultă în valoarea n)

Tot pe baza analogiei că o matrice este „un vector de vectori”, inițializarea unei matrici se scrie astfel:

```
int a[2][3]={1,2,3},{4,5,6};           // 2 linii și 3 coloane
```

Adică se consideră un vector de 2 elemente (liniile), a cărei elemente sunt fiecare câte un vector de 3 elemente (coloanele). Fiecare coordonată începe de la 0, deci primul element în această matrice va fi la $a[0][0]=1$ și ultimul element va fi la $a[1][2]=6$.

Elementele unei matrici se accesează analog celor ale unui vector, doar că se trec ambele dimensiuni:

```
// pentru matricea inițializată în exemplul anterior
a[0][2] = 7;           // pune 7 în căsuța de la linia 0 și coloana 2 (anterior era 3)
printf("%d", a[1][1]); // afișează 5
scanf("%d", &a[1][0]); // citește un element în locul vechii valori (4)
```

Exemplu: Se cer de la tastatură m și n , fiecare mai mici decât 10. Se cer apoi elementele unei matrici $a[m][n]$. Să se afișeze dacă matricea a are toate elementele pozitive.

```
#include <stdio.h>

int main()
{
    int a[10][10];           // rezerva spatiu pentru nr. maxim de elemente
    int i, j, m, n;
    printf("m=");
    scanf("%d", &m);
    printf("n=");
    scanf("%d", &n);
    //introducerea matricei
    for (i = 0; i < m; i++){           // i va itera liniile
```

```

for (j = 0; j < n; j++){           // j va itera coloanele
    printf("a[%d][%d]=", i, j);
    scanf("%d", &a[i][j]);
}
}

// algoritm care testeaza daca toate elementele indeplinesc o anumita conditie
int toate=1;                       // initial se presupune ca toate elementele indeplinesc conditia
for (i = 0; i < m; i++){
    for (j = 0; j < n; j++){
        if (a[i][j] < 0){
            toate=0;               // s-a gasit un element care nu indeplineste conditia => presupunere initiala falsa
            break;                 // nu mai are rost sa se continue iterarea
        }
        if (!toate){               // daca deja s-a gasit cel putin un element care nu indeplineste conditia, nu mai are rost
sa se itereze in continuare
            break;
        }
    }
}

if (toate){
    printf("matricea are toate elementele pozitive");
}else{
    printf("matricea are si elemente negative");
}
return 0;
}

```

Aplicația 8.1: Se citește de la tastatură un număr natural $n \leq 10$. Se citesc apoi elementele întregi ale unei matrici $a[n][n]$. Să se afișeze dacă matricea „a” este o matrice unitate. Notă: o matrice unitate are pe diagonala principală 1 și în rest 0.

Pentru a se afișa o matrice cu coloanele una sub alta, este necesar ca acestea să aibă aceeași lățime. Pentru a se asigura aceasta, se poate folosi în **printf** un specificator de număr de caractere, de exemplu „%3d”. Aceasta înseamnă că numărul respectiv va fi afișat pe 3 caractere, iar dacă acesta are mai puțin de 3 cifre, în fața lui se vor pune spații pentru completare. Pentru numere cu virgulă se poate folosi „%5.2g” pentru a se specifica faptul că se dorește afișarea cu 2 cifre zecimale. În acest caz numărul din fața punctului înseamnă numărul total de cifre, incluzând punctul, nu doar numărul de cifre din fața punctului.

Exemplu: Se cer de la tastatură m și n , fiecare mai mici decât 10. Să se creeze o matrice în care la fiecare poziție să fie media aritmetică a indecșilor acelei poziții și să se afișeze matricea.

```

#include <stdio.h>
int main()
{
    float a[10][10];
    int i, j, m, n;
    printf("m=");
    scanf("%d", &m);
    printf("n=");
    scanf("%d", &n);

    //initializare matrice
    for (i = 0; i < m; i++){

```

```

    for (j = 0; j < n; j++){
        a[i][j] = (i + j) / 2.0;
    }
}

//afisare
for (i = 0; i < m; i++){
    for (j = 0; j < n; j++){
        printf("%5.2g", a[i][j]);
    }
    printf("\n");
}
return 0;
}

```

Aplicația 8.2: Se citesc de la tastatură m și n , fiecare mai mici decât 10. Să se creeze o matrice în care la fiecare poziție să fie maximul indecșilor acelei poziții și să se afișeze matricea.

Matricile se pot folosi și pentru baze simple de date. De exemplu să presupunem că avem maxim m studenți, fiecare cu maxim n note. Se cere să se calculeze pentru fiecare student media sa. Am putea crea o matrice în care pe fiecare linie să fie notele unui student. În acest caz problema este că unii studenți pot avea mai puțin de n note și atunci nu trebuie luate în considerare toate valorile de pe acele linii din matrice.

Dacă avem matrici în care nu toate elementele sunt folosite, se pot folosi mai multe metode pentru a diferenția valorile din matrice:

1. se inițializează toate elementele cu o valoare care nu influențează rezultatul
2. se setează elementele nefolosite cu o valoare care are rol de indicator că acea poziție nu este folosită (de exemplu -1)
3. se folosește un vector auxiliar în care se setează pentru fiecare linie numărul de elemente folosite

```

#include <stdio.h>

int main()
{
    float a[10][10], suma;
    int i, j, m, n, k;
    printf("nr studenti=");
    scanf("%d", &m);
    printf("nr maxim note=");
    scanf("%d", &n);

    // initializare matrice cu o valoare care are rol de indicator ca acea locatie nu este folosita
    for (i = 0; i < m; i++){
        for (j = 0; j < n; j++){
            a[i][j] = -1;
        }
    }

    for (i = 0; i < m; i++){                // se itereaza pentru fiecare student
        printf("notele studentului %d\n", i);
        for (j = 0; j < n; j++){            // introducere note; notele unui student se termina la introducerea -1
            printf("nota: ");
            scanf("%g", &a[i][j]);
        }
    }
}

```

```

        if (a[i][j] == -1)break;
    }
}

for (i = 0; i < m; i++){
    k = 0;
    suma = 0;
    for (j = 0; j < n; j++){
        if (a[i][j] != -1){
            suma += a[i][j];
            k++;
        }else{
            break;
        }
    }

    if (k == 0){
        printf("studentul %d nu are note\n", i);
    }else{
        printf("studentul %d are media %g\n", i, suma / k);
    }
}

return 0;
}

```

Vectori și pointeri în contextul matricilor

Așa cum s-a discutat anterior, de fapt o matrice este un vector de vectori. Compilatorul va amplasa în memorie elementele matricii în ordinea liniilor, fără niciun spațiu între linii, conform figurii următoare (pentru o matrice $a[M][N]$):

$a[0][0]$	$a[0][1]$...	$a[0][N-1]$	$a[1][0]$	$a[1][1]$...	$a[1][N-1]$...	$a[M-2][0]$	$a[M-2][1]$...	$a[M-2][N-1]$	$a[M-1][0]$	$a[M-1][1]$...	$a[M-1][N-1]$
linia 0				linia 1				...	linia M-2				linia M-1			

Astfel, adresa de memorie a elementului $a[i][j]$, care este $\&a[i][j]$, se poate scrie ca fiind:

$$\&a[i][j] = \&a[0][0] + i*N + j$$

adică este adresa de început a matricii, la care se adaugă numărul de elemente $i*N$ conținut în liniile anterioare și în final se mai adaugă și numărul de elemente j până la indexul coloanei curente.

Atenție: literele M și N (cu majuscule) se referă la numărul maxim d

Folosind pointeri, putem de exemplu să preluăm adresa unui element și să iterăm elementele vecine cu el, așa cum s-a discutat la pointeri:

```

// initializeaza cu 0 toate elementele din matrice
int *p;
// expresia &a[0][0] are tipul int* deoarece este adresa unui element de tip int
// &a[m][0] este adresa primei locatii de memorie de dupa matrice; se mai poate folosi si &a[m-1][n]
for(p=&a[0][0] ; p<&a[m][0] ; ++p) *p=0;

```

Folosind acest mod de adresare a elementelor unei matrici, devine posibil să folosim la matrici funcțiile care operează cu vectori, cum ar fi sortare, min/max, căutare, etc.

Aplicații propuse

Aplicația 8.3: Se citește de la tastatură un număr $n \leq 10$ și un număr $m \leq 20$. Se cere să se scrie un program care generează o matrice cu m linii și n coloane în care sunt puse toate numerele de la $1, 2, 3, \dots, m \cdot n$. Se vor utiliza pointeri.

Aplicația 8.4: Se citește de la tastatură o matrice cu $m \leq 50$ linii și $n \leq 50$ coloane. Să se interschimbe elementele de deasupra diagonalei principale cu elementele de sub diagonala principală, elementele de pe diagonala principală rămânând la fel. Utilizați pointeri. De exemplu din matricea :

1 2 3

4 5 6

7 8 9

Va deveni:

1 4 7

2 5 8

3 6 9

Aplicația 8.5: Se citește de la tastatură o matrice cu $m \leq 20$ linii și $n \leq 20$ coloane. Să se memoreze într-un vector suma tuturor elementelor de pe fiecare linie și într-un alt vector, suma tuturor elementelor de pe fiecare coloană. Să se afișeze cei doi vectori. Se vor utiliza pointeri atât pentru vectori cât și pentru matrici.

Aplicația 8.6: Se citește un număr $n < 10$ de persoane, fiecare persoană fiind identificată printr-un număr între $0..n-1$. O persoană poate fi prietenă cu oricare alte persoane. După ce s-a citit n , se vor citi pentru fiecare persoană prietenii ei pe rând. Să se afișeze pentru fiecare persoană câți prieteni are.

Aplicația 8.7: Se citește un număr $n < 10$ de orașe și apoi pentru fiecare 2 orașe se citește distanța directă dintre ele. Dacă distanța este 0, înseamnă că între acele orașe nu există drum direct. Să se afișeze perechea de orașe cele mai apropiate între ele în mod direct.

Aplicația 8.8: Se citesc numerele m, n, p fiecare mai mici decât 10, apoi se citesc matricile $a[m][n]$ și $b[n][p]$. Se cere să se calculeze matricea „ c ” care rezultă din înmulțirea matricilor a cu b și să se afișeze.

Aplicația 8.9: Se citește un număr $n \leq 10$. Se cere să se inițializeze o matrice $a[n][n]$ cu 1 pe diagonale și cu 0 în rest. Să se afișeze matricea.

Aplicația 8.10: Se citește un număr $n \leq 10$ de produse și pentru fiecare produs vânzările lui trimestriale pe un an (4 valori). Se cere să se afișeze care produse au înregistrat o creștere continuă a vânzărilor în decurs de un an.

Exemple de probleme date la examene

Subiect 8.1: Se citesc de la tastatură două numere n și m , $n \geq m$. Pe urmă se citesc două matrici pătratice de dimensiuni n , respectiv m , conținând numere întregi. Afișați toate pozițiile unde a doua matrice apare în prima matrice. Afișarea se va face pe ecran sub forma unor perechi de forma (linie, coloană). Dacă a doua matrice nu apare în prima matrice, se va afișa textul “Nu apare” (examen PC, 2008).

Spre exemplu dacă se introduce:

5 3

1 0 1 1 1

0 1 0 1 1

1 0 1 1 1

0 1 0 1 1

1 0 1 1 1

0 1 1
1 0 1
0 1 1

se va afișa pe ecran:

(0,1) (2,1)

Subiect 8.2: Se citește de la tastatură un număr n și o matrice pătratică de dimensiune n conținând litere din alfabet. Pe urmă se citește de la tastatură un cuvânt. Verificați dacă respectivul cuvânt se poate construi parcurgând literele matricii pe verticală în sus sau în jos sau pe orizontală spre stânga sau spre dreapta. Afișați toate pozițiile de unde trebuie începută parcurgerea, precum și sensul de parcurgere necesar pentru a construi cuvântul (examen PC 2008).

Spre exemplu, dacă de la tastatură se introduce:

5
e r e m a
h e r e b
b m e r e
b a m e r
a e m r e

mere

pe ecran se va afișa:

De la linia 0 coloana 3 spre stânga.

De la linia 0 coloana 3 în jos.

De la linia 2 coloana 1 spre dreapta.

De la linia 3 coloana 2 în sus.

Nu are importanță ordinea în care sunt afișate soluțiile găsite. Dacă nu există soluție, se afișează "Nu există soluție".