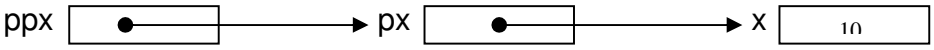
12. Tablouri şi pointeri (2).

12.1. Pointeri la pointeri.

Adresa unei variabile pointer va fi de tip pointer către pointer (pointer dublu)
Să considerăm definiţiile:

```
int x=10, *px=&x, **ppx=&px;
```

care corespund situaţiei:



pentru a obţine valoarea 10 putem folosi **x, *px** sau ****ppx**.
O funcţie care interschimbă doi pointeri are forma:

```
void pschimb(int **pa, int **pb)
{ int *ptemp;
  ptemp=*pa;
  *pa=*pb;
  *pb=ptemp;
}
```

cu apelul:

```
int *px, *py;
pschimb(&px, &py);
```

Deoarece un tablou este accesat printr/un pointer, tablourile de pointeri pot fi accesate cu pointeri dubli:

```
char *a[10];
char **pa;
pa = a;
```

Funcţia de afişare a şirurilor de caractere adresate de un tablou de pointeri poate fi rescrisă ca:

```
void afisare(char **tp, int n)
{ while(n--)
    printf("%s\n",*tp++);
}
```

12.2. Tablouri multidimensionale.

Un tablou cu mai multe dimensiuni se defineşte prin:

```
tip nume[d1][d2]…[dn];
```

Referirile la elemente unui tablou cu mai multe dimensiuni se fac folosind variabile indexate de forma: **nume[i1][i2]…[in],** în care **0<=ik<=dk-1**
Un tablou cu **n** dimensiuni poate fi considerat ca un tablou cu o dimensiune având ca elemente tablouri cu **n-1** dimensiuni.
Elementele tabloului ocupă o zonă continuă de memorie de:
d1 x d2 x…x dn x sizeof(T) octeţi.
Adresa în memorie a unui element **a[i1][i2]…[in]** este dată de funcţia de alocare:

```
&a[i1][i2]…[in]=a+sizeof(T)*[i1*d2*…*dn+i2*d3*…*dn+…+in]
```

În cazul vectorilor: **`&a[i]=a+sizeof(T)*i,`** ceea ce ne permite ca la declararea unui parametru vector să nu specificăm dimensiunea tabloului.
În cazul matricilor, compilatorul le transformă în vectori:

`&a[i][j]=a+sizeof(T)*(i*C+j)`

Şi în cazul matricelor, ca şi la vectori putem înlocui indexarea prin operaţii cu indici şi avem:

`a[i][j] = (*(a+i)[j]=*(*(a+i)+j)`

La transmiterea unui tablou multidimensional ca parametru al unei funcţii vom omite numai prima dimensiune, celelalte trebuind să fie specificate.
Prin urmare, prototipul unei funcţii de afişare a unei matrici având **`l`** linii şi **`c`** coloane nu poate fi scris ca:

```
void matprint(int a[][], int l, int c);
```

ci:

```
void matprint(int a[][DMAX], int l, int c);
```

în care **`DMAX`** este numărul de coloane al matricii din apel, ceeace ne limitează utilizarea funcţiei numai pentru matrici cu **`DMAX`** coloane!
Vom reda generalitate funcţiei de afişare a matricilor, declarând matricea parametru ca un vector de pointeri:

```
void matprint(int (*a)[], int l, int c)
{ int i,j;
  for(i=0;i<l;i++)
  { for (j=0;j<c;j++)
      printf("%4d",((int*)a)[i*c+j]);
    printf("\n");
  }
}
```

Problema transmiterii matricilor ca parametri poate fi evitată, dacă le linearizăm, transformându-le în vectori. În acest caz, în locul folosirii a doi indici i şi j vom folosi un singur indice:

```
k=i*C+j
```

Exemplul 30: ***Scrieţi o funcţie pentru înmulţirea a două matrici*** **`A`** ***şi*** **`B`** ***având*** **`mxn`**, ***respectiv*** **`nxp`** ***elemente.***

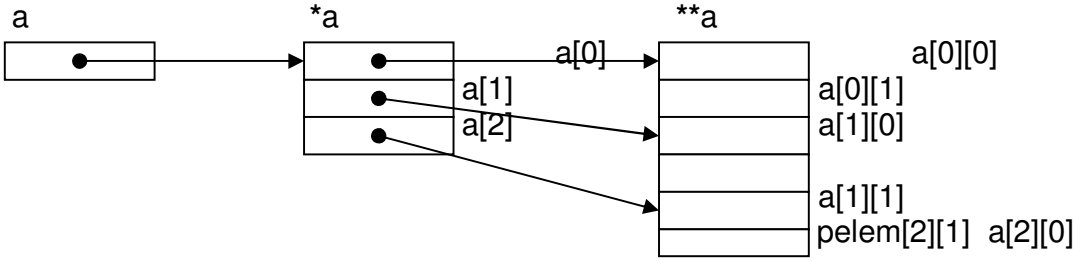
Matricea produs va avea **`mxp`** elemente, care vor fi calculate cu relaţia:

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj}$$

```
void matprod(int m,int n,int p,double A[],double B[], double C[])
{ int i, j, k, ij;
  for (i=0; i<m; i++)
    for (j=0; j<p; j++) {
      ij=i*p+j;
      for (k=0; k<n; k++)
        C[ij]=C[ij]+A[i*n+k]*B[k*p+j];
    }
}
```

Soluţia propusă nu ne permite să acceesăm elementele matricelor folosind 2 indici.
Am putea înlocui matricea printr-un vector de pointeri la liniile matricei.

***Exemplul 31*: Definiţi o funcţie care alocă dinamic memorie pentru o matrice având `lin` linii şi `col` coloane.**




```
double **alocmat(int lin, int col)
{  double **a;
   int i;
   a=(double**)calloc(lin, sizeof(double*));
   if(a==(double**)NULL) return NULL;
   for(int i=0; i<lin; i++){
     a[i]=(double*)calloc(lin, sizeof(double);
     if(a[i]==(double*)NULL) return NULL;
   }
  return a;
}
```

Eliberarea memoriei se face în ordine inversă: mai întâi se eliberează memoria ocupată de elementele din liniile matricei şi apoi pointerii la linii.

```
void elibmat(double **a, int lin){
  for(int i=0; i<lin; i++)
    free(a[i]);
  free(a);
}
```

12.3. Probleme propuse.

1. Să se construiască un patrat magic de dimensiune **n** (cu **n** impar), adică o matrice cu **n** linii şi **n** coloane având elemente numerele naturale $1,2,\ldots,n^2$ astfel încât sumele elementelor pe linii, pe coloane şi pe cele două diagonale să fie identice.

2 .Să se stabilească dacă există elemente comune tuturor liniilor unei matrici date. Se vor afişa câte asemenea elemente sunt, care sunt acestea şi apoi se va indica ce poziţie ocupă acestea în fiecare linie.

3. O matrice pătrată a are **n** linii şi **n** coloane. Cele două diagonale determină patru zone notate 1,2,3,4 care nu includ elementele de pe diagonale.
Să se calculeze mediile geometrice ale elementelor pozitive din zonele 1 şi 2. Dacă media nu se poate calcula, se va afişa un mesaj corespunzător.
Să se calculeze procentajul de elemente strict pozitive din zona 3 şi numărul de elemente divizibile

cu 5 din zona 4.Dacă nu există elemente cu proprietăţile cerute se va afişa un mesaj corespunzător.

4. Se dau două matrici **A** şi **B** pătrate, de ordin **n**. Să se stabilească dacă cele două matrici sunt sau nu una inversa celeilalte.În acest scop se creează matricea produs **C=A*B** şi se verifică dacă aceasta este matricea unitate.

5. Dintr-o matrice **A**, având n linii şi n coloane să se afişeze liniile care reprezintă şiruri ordonate crescător şi coloanele care reprezintă şiruri ordonate descrescător.

6. O matrice **a** are **p** linii şi **q** coloane. Să se creeze o nouă matrice **b**, din matricea a, exceptând liniile şi coloanele la intersecţia cărora se află elemente nule.Se vor utiliza doi vectori în care se vor marca liniile, respectiv coloanele care nu vor apare în **b**.

7. Se consideră o matrice **A** cu **p** linii şi **q** coloane, de elemente reale. Să se creeze pe baza acesteia o nouă matrice **B** având **m** coloane cu elementele pozitive din **A** şi un vector **C** cu elementele negative din matricea **A**.

8. Se dă o matrice **A** pătrată, cu **n** linii şi **n** coloane. Să se facă schimbările de linii şi de coloane astfel încât elementele diagonalei principale să fie ordonate crescător.

9. Să se calculeze coeficienţi polinomului Cebâşev de ordinul n, pornind de la relaţia de recurenţă

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \qquad k > 2$$
$$T_0(x) = 1, \quad T_1(x) = x$$

obţinând în prealabil relaţii de recurenţă pentru coeficienţi.

$$\bar{x}^T \bar{y} = \sum_{i=0}^{n-1} x_i y_i$$

10. a) Să se definească o funcţie care calculează produsul scalar a doi vectori, adică:

b) Să se definească o procedură care calculează produsul diadic a doi vectori:

$$\bar{x}\bar{y}^T = \begin{bmatrix} x_0y_0 & x_0y_1 & \cdots & x_0y_{n-1} \\ x_1y_0 & x_1y_1 & \cdots & x_1y_{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n-1}y_0 & x_{n-1}y_1 & \cdots & x_{n-1}y_{n-1} \end{bmatrix}$$

c) Să se scrie un program care citeşte: un număr întreg **n (n <= 10),** o matrice pătrată **A** cu **n** linii şi coloane şi doi vectori **u** şi **v** cu câte **n** componente şi calculează matricea **B**, în care:
B = A - u.v'/u'.v

11. Să se scrie un program care citeşte un număr întreg n şi o matrice pătrată **A** cu **n** linii şi coloane şi afişează numerele liniilor având în prima poziţie elementul minim şi în ultima poziţie - elementul maxim din linie. Se vor afişa de asemenea numerele coloanelor având în prima poziţie elementul maxim şi în ultima elementul minim.

12. Să se realizeze un program care simulează jocul "viaţa", adică trasează populaţia unei comunităţi de organisme vii, prin generarea de naşteri şi morţi timp de **G** generaţii.

Comunitatea de organisme este descrisă printr-o matrice cu **N** linii şi **N** coloane, fiecare element reprezentând o celulă care poate fi vidă sau poate conţine un organism. Fiecare celulă din reţea, exceptându-le pe cele de la periferie are 8 vecini.

Naşterile şi morţile de organisme se produc simultan la începutul unei noi generaţii. Legile genetice care guvernează creşterea şi descreşterea populaţiei sunt:

- fiecare celulă vidă care este adiacentă la 3 celule ocupate va da naştere în următoarea generaţie la un organism
- fiecare celulă care conţine un organism ce are numai un vecin sau niciunul,va muri la începutul generaţiei următoare(datorită izolării)
- orice organism dintr-o celulă cu patru sau mai mulţi vecini în generaţia prezentă va muri la începutul generaţiei următoare(datorită suprapopulaţiei).

13. Să se scrie în C:
a) o funcţie care verifică dacă 2 linii date **i** şi **j** dintr-o matrice pătrată (**nxn**) sunt identice sau nu.
b) o funcţie care afişează numerele liniilor şi coloanelor dintr-o matrice pătrată, unde se află elemente nule (zero).
c) un program care citeşte o matrice pătrată cu maxim 30 de linii şi coloane de numere întregi, verifică dacă există sau nu 2 linii identice în această matrice, folosind funcţia de la punctul a). Dacă toate liniile sunt distincte, atunci se afişează poziţia tuturor elementelor nule din matrice, folosind funcţia de la punctul b)

14. Să se înmulţească două matrici utilizând o funcţie pentru calculul produsului scalar a doi vectori.

15. Se dă o matrice **A** având **L** linii şi **C** coloane (**L,C≤10, C>3**) de elemente întregi.
Să se afişeze liniile în care există cel puţin trei elemente având minim cinci divizori nebanali. Se va defini şi utiliza o funcţie care stabileşte câţi divizori nebanali are un număr dat.

16.Să se definească o funcţie care calculează diferenţa între elementul maxim şi elementul minim ale unei linii date dintr-o matrice.

Să se scrie un program care citeste: numerele naturale **l** şi **c (l,c ≤ 10)**, valoarea reală **eps** şi matricea **A** având **l x c** elemente şi afişează liniile din matrice care au diferenţa între extreme inferioară valorii **eps**.

17. Intr-o matrice dată **A** cu **L** linii şi **C** coloane să se permute circular dreapta fiecare linie **i** cu **i** pozitii. Se va utiliza o funcţie care permută circular dreapta cu o poziţie componentele unui vector.

18. Pentru o matrice dată **A** cu **L** linii şi **C** coloane să se afişeze toate cuplurile **(i,j)** reprezentând numere de linii având elementele respectiv egale. Se va defini şi utiliza o funcţie care stabileşte dacă doi vectori sunt sau nu egali.

19. Se dau două matrici **A** şi **B** având n linii şi coloane fiecare. Să se stabilească dacă una dintre ele este sau nu inversa celeilalte. Se va defini şi utiliza o funcţie pentru a înmulţi două matrici.

20. Un punct şa într-o matrice este un element maxim pe coloană şi minim pe linia pe care se află sau minim pe coloană şi maxim pe linia sa.Utilizând funcţii care verifică dacă un element este minim/maxim pe linia/coloana sa să se determine punctele în şa dintr-o matrice cu elemente distincte.

21. Doi vectori **x** şi **y** cu câte **n** componente fiecare, **(n <= 20)** se află în relaţia **x <= y** dacă x_i **<=** y_i,pentru **i:=0..n-1**

a) Să se definească o funcţie care primind ca parametri două linii **i** şi **k** ale unei matrici, stabileşte dacă acestea se află în relaţia **<=.**

b) Să se definească o funcţie care, primind ca parametri numerele a două linii dintr-o matrice calculează diferenţa lor, depunând rezultatul într-un vector.

c) Să se scrie un program care citeşte un număr întreg **n (n <= 20)** şi o matrice cu **n** linii şi coloane şi afişează pentru toate perechile de linii care nu se găsesc în relatia **<=** diferenţa lor.

22.a) Să se definească o funcţie care stabileşte dacă o linie specificată a unei matrici este sau nu o secvenţă ordonată crescător.

b) Să se definească o funcţie care, pentru o linie specificată a unei matrici determină elementul minim şi elementul maxim din acea linie.

c) Se citeşte o matrice pătrată **A** cu **n** linii şi coloane (**n <= 10**). Să se afişeze pentru fiecare linie, care nu reprezintă un şir de valori crescătoare: numărul liniei, elementul maxim şi elementul minim.

Indicaţie: O linie **i** dintr-o matrice reprezintă o secvenţă ordonată crescător dacă:
$A_{i,j}$ **<=** $A_{i,j+1}$, pentru **j= 1:n-1**

21. Să se definească o funcţie care stabileşte dacă doi vectori daţi ca parametri sunt sau nu ortogonali.

Să se scrie un program care citeşte o matrice pătrată cu **n** linii şi coloane şi stabileşte dacă matricea este sau nu ortogonală pe linii, şi în caz afirmativ calculează matricea inversă. Se ştie că pentru o matrice ortogonală, matricea inversă se obţine transpunând matricea dată şi impărţind fiecare coloană cu norma euclidiană a ei (radicalul produsului scalar al coloanei cu ea însăşi). O matrice este ortogonală, dacă oricare două linii diferite sunt ortogonale.