FUNCŢII DE INTRARE – IEŞIRE CU FORMAT. TABLOURI BIDIMENSIONALE

1. Funcţiile scanf(...) şi printf(...)

Introducerea datelor se face în mod normal de la tastatura (stdin), iar afişarea rezultatelor se face pe monitor (stdout). Aceste simboluri sunt definite în STDIO.H.

stdin	dispozitivul periferic de intrare standard;
stdout	dispozitivul de ieşire standard;
stderr	dispozitivul de afişare a mesajelor de eroare (numai monitorul);
stdaux	dispozitivul de comunicaţii seriale (implicit COM1);
stdprn	dispozitivul de imprimare standard (implicit LPT1);

1.1. Funcţia printf(...)

Funcţia printf(...) scrie ieşirea formatată pe ecran. Are prototipul în STDIO.H.

```
int printf(const char * format [, argument, ...] );
```

Funcţia printf(...) returnează numărul de octeţi scrişi sau EOF (-1) în caz de eroare.

Specificatorii de format au următoarea formă generală:

%[flags][lăţime][.prec] tip_car

Fiecare specificator de conversie începe cu semnul procent (**%**) după care urmează în ordine:

- **flags**: o secvenţă opţională de caractere de control (cadrarea ieşirii la dreapta / stânga, afişarea zerourilor nesemnificative, prefixe octale sau hexazecimale, punctul zecimal, semn). Poate avea valorile:

  - aliniere la stânga, completează la dreapta cu blank-uri; dacă nu se specifică, se aliniază rezultatul la dreapta şi completează la stânga cu zero-uri sau blank-uri;

  + pune semn + sau - ; are prioritate faţă de blank dacă sunt ambele valori sunt prezente;

  blank (spaţiu) dacă numărul este pozitiv se pune blank în faţă, dacă e negativ se păstrează minusul.

- **lăţime**: un specificator opţional pentru numărul minim de caractere ce se vor afişa, completând la nevoie cu spaţii sau zero-uri. Poate avea una din valorile:

n număr întreg (vor fi afişate cel puţin n caractere, completându-se după caz cu spaţii);

0n vor fi afişate cel puţin n caractere, dar se completează cu zerouri.

- **prec:** un modificator opţional pentru afişare număr maxim de caractere sau de poziţii zecimale. Pentru întregi reprezintă numărul minim de digiţi ce trebuie afişaţi. Poate fi:

neprecizat: La afişare, precizia este cea implicită, adică

⇒ 1 – pentru tipurile d, i, o, u, x, X;

⇒ 6 – pentru e, E, f (6 poziţii zecimale implicit);

⇒ toate cifrele semnificative pentru G şi g;

⇒ afişează toate caracterele până la primul caracter diferit de NULL pentru tipul s; nu are efect la tipul c

0 pentru d, i, o, u, x precizia este cea implicită
pentru e, E, f nu e afişat simbolul de punct zecimal

n sunt afişate n caractere sau n poziţii zecimale

Dacă este cazul ieşirea poate fi trunchiată sau rotunjită.

Deci când se specifică o precizie **n**, efectul este:

- pentru d, i: sunt afişaţi cel puţin n digiţi;
- pentru o, u, x, X: dacă argumentul de intrare are un număr de digiţi < n, ieşirea se va completa cu zerouri (la stânga) până la atingerea preciziei; dacă argumentul de intrare are număr de digiţi > n, ieşirea nu va fi trunchiată
- pentru f, e, E: vor fi afişaţi n digiţi după punctul zecimal, iar ultimul digit va fi rotunjit (lipsă sau adaos după cum este <5 sau >5).
- pentru g, G: sunt afişaţi primii n digiţi semnificativi.

Observaţie: nu se va afişa nimic în cazul următor (toate condiţiile îndeplinite):

- precizia este pusă explicit zero
- tipul este unul dintre d, i, o, u, x
- valoarea de afişat este nulă

- **tip_car** (caracterul care indică tipul conversiei)

Caracter	Valoare de intrare aşteptată	Ieşire formatată
d	Întreg	Întreg zecimal cu semn
i	Întreg	Întreg zecimal cu semn
o	Întreg	Întreg octal fără semn
u	Întreg	Întreg zecimal fără semn
x	Întreg	Întreg hexazecimal fără semn (cu litere mici)
X	Întreg	Întreg hexazecimal fără semn (cu litere mari)
f	Real	Valoare cu semn de forma [-]dddd.dddd
e	Real	Valoare cu semn de forma [-]d.dddd e[+/-]ddd
E	Real	Idem; se foloseşte E pentru exponent
g	Real	Valoare cu semn în forma e sau f (se alege forma ce ocupa număr minim de poziţii)
G	Real	Idem cu g; cu E pentru exponent
%	-	Tipăreşte simbolul %
p	Pointer	Tipăreşte argumentul ca xxxx:yyyy (segment :offset) sau numai în forma yyyy

Să detaliem :

- litera **d** afişează valori de tip întreg

```
printf("*%10d*",123);          /* afişează: *       123*       */
printf("*%-10d*",123);         /* afişează: *123      *        */
printf("*%010d*",123);         /* afişează: *0000000123* */
```

- litera **o** afişează în octal date de tip int sau unsigned (întreg şi întreg fără semn)

```
printf("*%10o*",123);          /*      afişează: *       173*
                                (deoarece 173 în baza 8 este 123 //zecimal)      */
```

- literele **x, X** afişează în hexazecimal date de tip int sau unsigned (întreg şi întreg fără semn)

x se foloseşte pentru litere mici a ... f

X se foloseşte pentru litere mari A ... F

```
printf("*%10x*,123);           /*      afişează: *       7b"          */
```

- litera **u** la fel cu d; se foloseşte pentru conversia din unsigned în zecimal
- litera **l** poate însoţi una din literele d, o, x, X, u

ld conversia din long int (întreg lung) în zecimal

lu conversia din long unsigned (întreg lung fără semn) în zecimal

lo conversia din long sau long unsigned în octal

lx conversia din long sau long unsigned în hexazecimal

lX la fel u **lx**, dar se folosesc litere mari

Pentru tipul de dată long long se foloseşte prefixul **ll**.

- litera **f** afişează numere reale, de tip float. Pentru numere de tip double se foloseşte **lf**, iar pentru long double **Lf**.

Aceste numere pot avea o parte întreagă şi o parte fracţionară sau numai parte întreagă. Numărul de zecimale e definit de precizia indicată în specificatorul de format. Dacă nu e precizat numărul de zecimale, implicit se afişează 6 zecimale. Ultima cifră e rotunjită prin adaos sau lipsă după cum cifra este >=5 sau <5

Exemple:

Valoare	Specificator	Rezultat afişat
3.14159265	%5f	3.14153 (rotunjire adaos la 6 zecimale)
123.672	%7f	123.672000 (completare cu 0 până la 6 zecimale)
3.14159265	%7.2f	3.14 (rotunjire lipsă până la 2 zecimale)
123.672	%10.1f	123.7 (rotunjire adaos până la o zecimală)
-123.672	%10.1f	-123.7 (idem)
3.14159265	%10.0f	3 (rotunjire lipsă, nici o zecimală)
123.672	%10.0f	124 (rotunjire adaos, nici o zecimală)

- literele **e, E** afişează un număr real (de tip float sau double) sub forma: p_int.p_fract exp sau p_int exp;

Exemple:

Număr	Specificator	Rezultat afişat
3.14159265	%e	3.141593e+00
123.672	%e	1.236720e+02
123.672	%.1E	1.2e+02
0.673	%E	6.730000E-01

- literele **g, G** funcţionează la fel ca **f** sau ca **e, E**. Se alege forma convenabilă pentru a afişa un număr minim de caractere; afişează 6 zecimale numai dacă acestea sunt semnificative; afişează punctul zecimal numai dacă este prezentă partea fracţionară. Se foloseşte g pentru e, G pentru E (desigur dacă s-a preferat forma e, E formei f). Tipul float are 6,7 zecimale; nu se recomandă afişarea unui număr mai mare de zecimale. Tipul double are 15 zecimale; alegerea afişării cu un număr mai mare de zecimale nu are sens;

- litera **L** poate însoţi literele f, e, E, g, G. Data care se afişează este de tip long double (real lung dublu). Lf este forma fără exponent, Le, LE forma cu exponent, iar Lg, LG forma mai convenabilă dintre f şi e(E)

Observaţii:

1). $+\infty$ şi $-\infty$ sunt afişate ca +INF şi -INF

2). Dacă rezultatul nu este număr (not_a_number) se afişează +NAN sau -NAN

3). La formatul %e sau %E se converteşte argumentul în forma [-]d.ddd...e[+/-]ddd cu o cifră înaintea punctului zecimal, un număr de cifre după punct egal cu precizia şi exponentul având cel puţin doua cifre.

4). La formatul %f se converteşte argumentul în forma [-]ddd.ddd..., cu număr de cifre de după punctul zecimal egal cu precizia indicată (poate fi zero).

5). La %g sau %G se elimină zerourile nesemnificative şi punctul zecimal apare numai dacă este nevoie. Argumentul este afişat în forma e sau f (pentru g), respectiv E (pentru G), depinde care formă e mai avantajoasă (mai scurtă)

1.2. Funcţia fprintf(...)

Această funcţie are prototipul în fişierul header **stdio.h**, de forma:

```
int fprintf(FILE *fisier, const char * format[, argument, ...] );
```

unde **fisier** este variabila care a fost asociată fişierului cu care se lucrează la deschiderea acestuia.

Afişarea pe monitor a unui mesaj se poate face şi prin intermediul funcţiei **fprintf** al cărui prim argument este **stdout**.

Exemplu: afişarea mesajului

Program pentru calcularea sumei a doua matrice

se poate face prin apelul funcţiei **printf** astfel:

printf("Program pentru calcularea sumei a doua matrice\n");

sau prin apelul funcţiei **fprintf** astfel:

fprintf(stdout, "Program pentru calcularea sumei a doua matrice\n");

1.3. Funcţia scanf(...)

Funcţia (cu prototipul tot în STDIO.H) citeşte şi formatează datele de intrare (citite de la tastatură). Are formatul:

```
int scanf(const char *format [,address,...] );
```

scanf(...) citeşte elementele de intrare (caracter cu caracter) de la tastatură, după care le formatează în conformitate cu specificatorul din format şi depune rezultatul la

adresa transmisă ca argument (după format). Trebuie sa fie acelaşi număr de specificatori de format şi de adrese ale elementelor de intrare. Dacă sunt mai puţine adrese, efectul este imprevizibil. Argumentele adresă în exces sunt ignorate. Funcţia începe scanarea după apăsarea tastei Enter (cu condiţia să se fi introdus minimum atâtea elemente câte adrese apar. Elementele introduse în plus rămân pentru citirile ulterioare).

Şirul de caractere format poate conţine:

a) caractere albe - whitespaces (spaţiul, tab - `\t', newline - `\n', carriage return – '\r')

Dacă scanf(...) întâlneşte un astfel de caracter în format, va citi (fără să le memoreze) toate spaţiile albe consecutive până la următorul caracter diferit din intrare.

b) caractere non whitespace (toate celelalte caractere ASCII fără "%")

Dacă scanf(...) întâlneşte un caracter ce nu este caracter alb în format, va citi (fără memorare) caracterul corespunzător (dacă în intrare este un caracter diferit, se încheie scanarea).

c) specificatori de format : **%[*][lăţime] tip_car**

Forma minimă a specificatorului de format: începe cu **%** şi se termină cu 1-2 litere ce definesc tipul conversiei)

lăţime - reprezintă număr maxim de caractere (n- întreg zecimal) ce urmează a fi citite. Sunt citite, convertite şi stocate la adrese, până la n caractere

tip_car - aceeaşi semnificaţie ca la printf(...). În tabelul de mai jos valorile posibile pentru acest parametru:

Tip_car	La intrare trebuie să fie
d	Întreg zecimal
D	Întreg zecimal
e, E	Real
F	Real
g, G	Real
o	Întreg octal
O	Întreg octal
i	Întreg zecimal, octal sau hexazecimal
I	Întreg zecimal, octal sau hexazecimal
u	Întreg zecimal fără semn
U	Întreg zecimal fără semn
x	Întreg hexazecimal
X	Întreg hexazecimal

s	Şir de caractere
c	Caracter

Toate argumentele trebuie să fie adrese ale unor variabile de tipul indicat (adresa este indicată prin prezenţa operatorului **&**).

Elementele din intrare se separă astfel:

- toate caracterele până la următorul spaţiu alb (exclusiv);
- toate caracterele până la primul care nu poate fi convertit conform formatului specificat;
- până la **n** caractere (specificat prin câmpul lăţime).

La folosirea formatului **"%c"** se citeşte următorul caracter, chiar dacă este caracter alb. Pentru a se sări la următorul caracter ce nu este alb şi a-l citi se recomandă să se folosească " %c" (spaţiul de dinainte de %c va face să se sară peste toate “spaţiile albe” intermediare). Dacă asteriscul (assignment suppression character) urmează semnului %, următorul element din intrare va fi scanat (conform tip_car din continuare), dar nu va fi asignat următorului argument adresă (nu se poate determina dacă operaţia s-a făcut cu succes sau nu). Puteţi întâlni pentru a sări peste spaţiile albe din intrare forma scanf("%*c").

Funcţia scanf(...) se termină la:

- tastarea combinaţiei de taste CTRL-Z pentru Windows sau CTRL-D pentru Linux
- la terminarea scanării elementului curent din intrare;
- dacă următorul element din intrare nu poate fi convertit conform formatului;
- dacă s-a atins limita indicată prin “lăţime”

Elementul la care se termină scanf(...) se va considera ca necitit şi va fi primul pentru următoarea operaţie de citire de la stdin. Funcţia returnează numărul elementelor din intrare ce au fost scanate, convertite şi memorate. Dacă se detectează EOF (s-a apăsat CTRL-D (Linux) sau CTRL-Z (Windows)), funcţia returnează valoarea -1. Datele se citesc efectiv după acţionarea tastei ENTER.

Pentru golirea bufferului tastaturii se poate folosi următoarea funcție:

```
void clean_stdin(void)
{
    int c;
    do {
        c = getchar();
    } while (c != '\n' && c != EOF);
}
```

Când se citesc şiruri de caractere (tablouri) nu se mai foloseşte operatorul **&** în faţă ca la variabile simple, deoarece numele unui tablou este el însuşi o adresă de memorie şi anume adresa primului element din tablou.

Să detaliem literele ce pot apare după %:

- litera **d**: citeşte întregi zecimali (date de tip întreg zecimal cu semn – **int** Exemplu 1:

```
int i1,i2,i3;
scanf("%2d%3d%2d",&i1,&i2,&i3);
```

La intrare avem:1234567, atunci: i1=12; i2=345; i3=67

Exemplu 2:

```
int n;
scanf("%d", &n);
```

La intrare: i23 atunci se returneaza 0, căci i nu corespunde formatului. Dacă era la intrare:23i atunci se citea 23.

- litera **o**: la fel ca d, dar se citeşte un întreg octal
- literele **x, X**: la fel cu d, dar se citeşte un întreg în forma hexazecimală
- litera **u**: pentru citire întregi zecimali, tip unsigned (fără semn, deci numere naturale)
- litera **f**: citire număr real simplă precizie (tip **float**)
- literele **e, E**: citire număr flotante simplă precizie în forma cu exponent
- literele **g, G**: citire număr flotante în forma f sau e(E)
- litera **l**: poate însoţi d, o, x, X, u, f.
  - ld, lo, lx, lX: data citită e convertită spre long
  - lu: data citită e convertită spre unsigned long
  - lf: data citită este convertită spre flotant dublă precizie (tipul double)
- literele **D, O, U, l**: pentru numere întregi de tipul long.
- litera **c**: se citeşte caracterul curent chiar dacă este alb. Exemplu:

```
scanf("%c", &var_c);
```

- litera **s**: începe cu caracterul care nu e alb şi se continuă până la alt caracter alb sau până se atinge lungimea maximă. La sfârşitul unui şir citit astfel se pune automat '\0' – marcajul de sfârşit de şir

Exemplu1:

```
....
char şir[2];
scanf("%1s",şir);          //citeşte un singur caracter, primul ce nu este alb
                           //memorează şi '\0' la sfârşit
....
```

Exemplu 2:

```
char tab1[10];
char tab2[10];
scanf("%2s%9s", tab1, tab2);
....
```

Să presupunem că se tastează la intrare şirul “necunoscut”. Atunci în **tab1** se va păstra "ne", iar în **tab2** "cunoscut", desigur terminate cu '\0'

1.4. Funcţia fscanf(...)

Funcţia (cu prototipul tot în STDIO.H) citeşte şi formatează datele de intrare. Are formatul:

```
int fscanf(FILE *fisier, const char *format [,address,...] );
```

unde **fisier** este fişierul din care se citesc datele.

În cazul în care fisier este stdin, datele se vor citi de la tastatură.

Astfel:

```
fscanf(stdin, "%d", &n);
```

citeşte de la tastatură valoarea variabilei **n**.

Tablouri multidimensionale

Aşa cum am văzut în laboratorul trecut, în limbajul **C** putem lucra cu tablouri. Definiţia:

```
int a[10];
```

se referă la un tablou de 10 elemente în care fiecare element este un întreg cu semn.

Limbajul permite şi folosirea de tablouri în care fiecare element este un tablou. Definiţia:

```
double mat[2][3];
```

se referă la un tablou cu două elemente, în care fiecare element este un tablou de trei elemente de tip real în dublă precizie. (Un astfel de tablou este, de fapt, o matrice cu două linii, iar fiecare linie are trei elemente de tip real în dublă precizie).

Tablourile care au ca elemente alte tablouri sunt tablouri multidimensionale,

Pentru cazul general

T nume[dim_1][dim_2]...[dim_n];

unde **nume** este numele unui tablou cu dim_1 elemente, în care fiecare din cele dim_1 este un tablou cu dim_2 elemente, în care fiecare din cele dim_2 elemente este un tablou ..., iar **T** este tipul elementelor din tabloul de dimensiune dim_n.

În problemele propuse din acest laborator se vor folosi tablouri bidimensionale (matrice).

TEMA 1

Problema 1.1

Să se scrie un program care calculează suma şi produsul a două matrice A şi B. Elementele matricelor sunt numere reale. Se vor scrie funcţii pentru: citirea unei matrice dintr-un fişier, afişarea unei matrice pe monitor (în formatul cunoscut de la matematică – pe linii), calculul sumei a două matrice şi calculul produsului a doua matrice. Matricele nu sunt neapărat pătratice. Alegerea operaţiei (suma sau produs de matrice) se va face de către utilizator la rularea programului. Pentru fiecare din cele două operaţii trebuie făcută validarea dimensiunilor matricelor astfel încât operaţia să poată fi făcută.

Afişarea unei matrice pe monitor se va face cu funcţia **fprintf** al cărui prim parametru va fi stdout.

Problema 1.2

Să se scrie o funcţie care calculează transpusa unei matrice pătratice. Se vor afişa matricea iniţială şi cea transpusă.

Să se verifice relaţia:

$$(A \cdot B)^T = B^T \cdot A^T$$

unde A, B sunt matrice pătratice, iar A^T este transpusa matricei A. Se vor afişa cele două matrice, transpusele lor, precum şi cei doi membri ai egalităţii.

Citirea se va face de la tastatură (cu funcţia **fscanf**, primul parametru fiind stdin), iar afişarea se va face pe monitor.

Problema 1.3

Pentru rezolvarea acestei probleme se vor folosi fişierul header şi fişierul cu funcţii scrise pentru Problema 1.1.

În Pădurea cu alune au case **n** pitici. Cămara fiecărei case are **m** rafturi, câte un raft pentru fiecare din cele **m** produse pe care piticii le pot cumpăra de la magazinul aflat la marginea pădurii. De la magazin poate cumpăra doar starostele piticilor şi numai în prima zi din lună. Pentru a putea face cumpărăturile necesare, la fiecare sfârşit de lună starostele piticilor trece pe la casa fiecărui pitic şi ia comanda acestuia, după care se duce la magazin şi face cumpărăturile necesare.

Toate produsele din magazin se vând la bucată.

Trebuie să se calculeze:

1) Care sunt stocurile de alimente ale piticilor după ce starostele s-a întors de la magazin?
2) Cât trebuie să plătească fiecare pitic dacă se cunosc preţurile pe bucată ale celor **m** produse existente în cămările piticilor?
3) Câte zile trebuie să muncească fiecare pitic pentru a achita nota de plată la magazin? Se presupune că piticii nu fac alte cheltuieli în afara celor pentru mâncare şi că pentru o zi de muncă fiecare pitic primeşte **G** galbeni. (Se va face rotunjirea la număr întreg de zile, de exemplu: dacă rezultă 2.34 zile, se va afişa atât valoarea exactă calculată – în cazul acesta 2.34 zile – cât şi rotunjirea – în acest caz 3 zile -.)

Date de intrare

Datele de intrare se citesc din fişierul **PITICI.IN** care are pe prima linie un număr real care reprezintă plata pentru o zi de lucru (**G**).

Pe următoarea linie se găsesc numărul **n** de pitici din pădure şi numărul **m** de rafturi din cămara fiecărui pitic.

Pe următoarele **n** linii se găsesc stocurile din cele **m** alimente pe care le are fiecare pitic date ca numere întregi pozitive câte **m** pe o linie separate de câte un spaţiu.

Pe ultima linie se găsesc preţurile celor **m** alimente necesare piticilor date ca numere reale separate de câte un spaţiu.

Date de ieşire

Rezultatele obţinute trebuie înscrise în fişierul **PITICI.OUT**, iar datele de intrare trebuie afişate pe monitor pentru a verifica citirea lor corectă.

TEMA 2

Problema 2.1

Pentru o matrice pătratică cu elemente numere reale se definesc următoarele norme:

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|$$ (maximul sumelor de pe fiecare linie)

$$\|A\|_1 = \max_j \sum_i |a_{ij}|$$ (maximul sumelor de pe fiecare coloană)

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2}$$

unde $|x|$ defineşte modulul numărului real x.

Să se calculeze normele definite mai sus. Se va realiza un dialog prin care să se aleagă opţiunea de prelucrare. Programul va fi realizat astfel încât să permită prelucrarea

mai multor seturi de date. Citirea se va face dintr-un fişier, iar scrierea rezultatelor se va face în alt fişier.

Problema 2.2

Se citeşte o matrice **A** de dimensiune **n** × **n**, cu **n** ≤ 20 (se va face validarea dimensiunii **n**).

Matricea poate fi prelucrată (alegerea este a utilizatorului) astfel încât:

a) elementele de pe prima linie să fie în ordine crescătoare
b) elementele de pe prima coloană să fie în ordine crescătoare
c) elementele de pe diagonala principală să fie în ordine crescătoare

Se va lua în considerare posibilitatea reluării programului cu alte date de test.

Trebuie scrise funcţii pentru: citire matrice, afişare matrice, interschimbare a două linii din matrice, interschimbare a două coloane dintr-o matrice, câte o funcţie pentru fiecare opţiune de sortare. Cele două funcţii de interschimbare vor primi ca parametri matricea, dimensiunea matricei, şi cei doi indecşi ai liniilor, respectiv coloanelor care se interschimbă.

TEMA 3

Problema 3.1

P3.1.a. Să se deducă relaţii de recurenţă pentru calcularea următoarelor matrice:

$T_i = \frac{A^i}{i!}$, cu $T_0 = I$ (unde A este o matrice pătratică de ordin **n**, iar I este matricea unitate de ordin **n**).

$$B_n = \sum_{i=0}^{n} T_i \text{, cu } B_0 = I \qquad (1)$$

P3.1.b. Să se scrie un program care

1). citeşte de la tastatură o matrice **A** de numere reale şi de dimensiune **n** (cu **n** linii şi **n** coloane), cu **n ≤ 20** şi afişează matricea citită.

2). Calculează matricea **B** cu expresia dată de relaţia (1)

3). Afişează matricea **B** calculată la punctul 2).

Programul va fi astfel construit încât să poate fi introduse mai multe seturi de date, iar valoarea dimensiunii **n** citite de la tastatură va trebui validată (dacă n ≤ 0 sau n ≥ 20 se va afişa un mesaj de eroare şi se va cere citirea unei noi valori (corecte)).

Date de test:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 95 & 116 & 138 \\ 214 & 264 & 312 \\ 334 & 410 & 487 \end{bmatrix}$$

şi

$$A = \begin{bmatrix} 1 & 5 & 9 & 0 \\ 0 & 3 & 14 & 7 \\ 21 & 43 & 17 & 0 \\ 11 & -3 & 24 & -15 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.2965e+004 & 2.6409e+004 & 2.2184e+004 & 2.4646e+003 \\ 1.9550e+004 & 3.9034e+004 & 3.7497e+004 & 4.2440e+003 \\ 4.3914e+004 & 9.6555e+004 & 7.9225e+004 & 1.3801e+004 \\ 2.5745e+004 & 4.9372e+004 & 3.4299e+004 & 5.1896e+002 \end{bmatrix}$$

Barem de notare:

1. Citire şi validare dimensiune matrice	**0,5**
2. Citire matrice	**1,0**
3. Afişare matrice	**1,0**
4. Calcul matrice **B**	**1,0**
4a). Adunarea a două matrice	**1,0**
4b). Înmulţirea a două matrice	**1,5**
4c). Înmulţirea unei matrice cu un scalar	**1.0**
5. Posibilitatea de reluare a programului	**1,0**
6. Calcul matrice **T** conform relației de recurență	**0,5**
7. Scriere corectă fişier header	**0,5**
8. Funcţia main	**1,0**
TOTAL	**10 p**

Problema 3.2

Un teren de formă dreptunghiulară este împărţit în parcele, dispuse pe **n** rânduri, pe fiecare rând fiind **m** parcele. Se măsoară altitudinea fiecărei parcele (se consideră că o parcelă are altitudine constantă). Altitudinile tuturor parcelelor se memorează într-un tablou cu **n** linii şi **m** coloane. Valoarea elementului de pe linia **i** şi coloana **j** a tabloului este altitudinea celei de a **j**-a parcele de pe linia **i**.

Să se afişeze coordonatele “parcelelor vârf”. O “parcelă vârf” are altitudinea strict mai mare decât a tuturor vecinilor săi.

Se va lua în considerare posibilitatea prelucrării mai multor seturi de date, la aceeaşi rulare a programului.

Problema 3.3

Se dă o matrice pătratică de ordin **n**. Se consideră că diagonalele împart matricea în patru zone: nord, sud, est, vest (elementele de pe diagonale nu fac parte din nici o zonă).

Să se afişeze matricea citită.

Să se calculeze suma elementelor din nord, produsul elementelor din sud, media aritmetică a elementelor din est şi media geometrică a elementelor din vest.

Să se determine imaginea în oglindă a matricii iniţiale şi să se afişeze.

Se va lua în considerare posibilitatea prelucrării mai multor seturi de date, la aceeaşi rulare a programului.