

FUNCTII DE INTRARE – IEȘIRE CU FORMAT. TABLOURI BIDIMENSIONALE

1. Funcțiile scanf(...) și printf(...)

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

stdin	dispozitivul periferic de intrare standard;
stdout	dispozitivul de ieșire standard;
stderr	dispozitivul de afișare a mesajelor de eroare (numai monitorul);
stdaux	dispozitivul de comunicații seriale (implicit COM1);
stdprn	dispozitivul de imprimare standard (implicit LPT1);

1.1. Funcția printf(...)

Funcția printf(...) scrie ieșirea formatată pe ecran. Are prototipul în STDIO.H.

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

Funcția printf(...) returnează numărul de octeți scriși sau EOF (-1) în caz de eroare.

Specificatorii de format au următoarea formă generală:

%[flags][lățime][.prec] tip_car

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

- **flags**: o secvență opțională de caractere de control (cadrarea ieșirii la dreapta / stânga, afișarea 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 și completează la stânga cu zero-uri sau blank-uri;

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

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

- **lățime**: un specificator opțional pentru numărul minim de caractere ce se vor afișa, completând la nevoie cu spații sau zero-uri. Poate avea una din valorile:

n număr întreg (vor fi afișate cel puțin n caractere, completându-se după caz cu spații);

0n vor fi afișate cel puțin n caractere, dar se completează cu zerouri.

- **prec**: un modificador opțional pentru afișare număr maxim de caractere sau de poziții zecimale. Pentru întregi reprezintă numărul minim de digiți ce trebuie afișați. Poate fi:

neprecizat: La afișare, precizia este cea implicită, adică

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

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

⇒ toate cifrele semnificative pentru G și g;

⇒ afișează 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 afișat simbolul de punct zecimal

n sunt afișate n caractere sau n poziții zecimale

Dacă este cazul ieșirea poate fi trunchiată sau rotunjită.

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

- pentru d, i: sunt afișați cel puțin n digiți;

- pentru o, u, x, X: dacă argumentul de intrare are un număr de digiți < n, ieșirea se va completa cu zerouri (la stânga) până la atingerea preciziei; dacă argumentul de intrare are număr de digiți > n, ieșirea nu va fi trunchiată

- pentru f, e, E: vor fi afișați n digiți după punctul zecimal, iar ultimul digit va fi rotunjit (lipsă sau adaos după cum este <5 sau >5).

- pentru g, G: sunt afișați primii n digiți semnificativi.

Observație: nu se va afișa nimic în cazul următor (toate condițiile îndeplinite):

- precizia este pusă explicit zero

- tipul este unul dintre d, i, o, u, x

- valoarea de afișat este nulă

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

Caracter	Valoare de intrare așteptată	leșire 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 folosește E pentru exponent
g	Real	Valoare cu semn în forma e sau f (se alege forma ce ocupa număr minim de poziții)
G	Real	Idem cu g; cu E pentru exponent
%	-	Tipărește simbolul %
p	Pointer	Tipărește argumentul ca xxxx:yyyy (segment :offset) sau numai în forma yyyy

Să detaliez :

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

```
printf("%10d",123);          /* afișează: *    123*    */
```

```
printf("%-10d",123);         /* afișează: *123   *    */
```

```
printf("%010d",123);         /* afișează: *000000123* */
```

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

```
printf("%10o",123);          /* afișează: *    173*
```

(deoarece 173 în baza 8 este 123 //zecimal) */

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

x se folosește pentru litere mici a ... f

X se folosește pentru litere mari A ... F

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

- litera **u** la fel cu d; se folosește pentru conversia din unsigned în zecimal

- litera **l** poate însoți 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

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

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

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

Aceste numere pot avea o parte întreagă și o parte fracționară 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 afișează 6 zecimale. Ultima cifră e rotunjită prin adaos sau lipsă după cum cifra este ≥ 5 sau < 5

Exemple:

Valoare	Specificator	Rezultat afișat
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** afișează un număr real (de tip float sau double) sub forma: p_int.p_fract exp sau p_int exp;

Exemple:

Număr	Specificator	Rezultat afișat
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** funcționează la fel ca **f** sau ca **e**, **E**. Se alege forma convenabilă pentru a afișa un număr minim de caractere; afișează 6 zecimale numai dacă acestea sunt semnificative; afișează punctul zecimal numai dacă este prezentă partea fracționară. Se folosește **g** pentru **e**, **G** pentru **E** (desigur dacă s-a preferat forma **e**, **E** forme **f**). Tipul float are 6,7 zecimale; nu se recomandă afișarea unui număr mai mare de zecimale. Tipul double are 15 zecimale; alegerea afișării cu un număr mai mare de zecimale nu are sens;

- litera **L** poate însoți literele f, e, E, g, G. Data care se afișează 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 și e(E)

Observații:

- 1). $+\infty$ și $-\infty$ sunt afișate ca +INF și -INF
- 2). Dacă rezultatul nu este număr (not_a_number) se afișează +NAN sau -NAN
- 3). La formatul %e sau %E se convertește argumentul în forma [-]d.ddd...e[+/-]ddd cu o cifră înaintea punctului zecimal, un număr de cifre după punct egal cu precizia și exponentul având cel puțin doua cifre.
- 4). La formatul %f se convertește 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 ne semnificative și punctul zecimal apare numai dacă este nevoie. Argumentul este afișat în forma e sau f (pentru g), respectiv E (pentru G), depinde care formă e mai avantajoasă (mai scurtă)

1.2. Funcția fprintf(...)

Această funcție are prototipul în fișierul header **stdio.h**, de forma:

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

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

Afișarea pe monitor a unui mesaj se poate face și prin intermediul funcției **fprintf** al cărui prim argument este **stdout**.

Exemplu: afișarea mesajului

Program pentru calcularea sumei a doua matrice

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

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

sau prin apelul funcției **fprintf** astfel:

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

1.3. Funcția scanf(...)

Funcția (cu prototipul tot în STDIO.H) citește și formatează datele de intrare (citite de la tastatură). Are formatul:

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

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

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

Șirul de caractere format poate conține:

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

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

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

Dacă scanf(...) întâlnește 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 : %[*][lățime] tip_car

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

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

tip_car - aceeași semnificație 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
l	Întreg zecimal, octal sau hexazecimal
u	Întreg zecimal fără semn
U	Întreg zecimal fără semn
x	Întreg hexazecimal
X	Întreg hexazecimal

s	Șir de caractere
c	Caracter

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

Elementele din intrare se separă astfel:

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

La folosirea formatului **"%c"** se citește următorul caracter, chiar dacă este caracter alb. Pentru a se sări la următorul caracter ce nu este alb și a-l citi se recomandă să se folosească **" %c"** (spațiul de dinainte de %c va face să se sară peste toate "spațiile 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 operația s-a făcut cu succes sau nu). Puteți întâlni pentru a sări peste spațiile albe din intrare forma `scanf("%*c")`.

Funcția `scanf(...)` se termină la:

- tastarea combinației de taste CTRL-Z;
- 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 "lățime"

Elementul la care se termină `scanf(...)` se va considera ca necitit și va fi primul pentru următoarea operație de citire de la `stdin`. Funcția returnează numărul elementelor din intrare ce au fost scanate, convertite și memorate. Dacă se detectează EOF (s-a apăsător CTRL-Z), funcția returnează -1. Datele se citesc efectiv după acționarea tastei ENTER . Pentru o golire totală a tuturor datelor rămase necitite în intrare se poate folosi funcția: `fflush(stdin);` //(cu prototipul în `STDIO.H`)

Când se citesc șiruri de caractere (tablouri) nu se mai folosește operatorul **&** în față ca la variabile simple, deoarece numele unui tablou este el însuși o adresă de memorie și anume adresa primului element din tablou.

Să detaliam literele ce pot apare după %:

- litera **c**: se citește caracterul curent chiar dacă este alb. Exemplu:

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

- litera **s**: începe cu caracterul care nu e alb și se continuă până la alt caracter alb sau până se atinge lungimea maximă. La sfârșitul unui șir citit astfel se păstrează automat '\0'
- marcajul de sfârșit de șir

Exemplu1:

```
....
char șir[2];
scanf("%1s",șir);           //citește un singur caracter, primul ce nu este alb
                             //memorează și '\0' la sfârșit
....
```

Exemplu 2:

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

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

- litera **d**: citește întregi zecimali (date de tip întreg zecimal cu semn – **int**), deci domeniul de valori este [- 32768, 32767].

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 citește un întreg octal
- literele **x, X**: la fel cu d, dar se citește 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 însoți 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.

1.4. Funcția fscanf(...)

Funcția (cu prototipul tot în STDIO.H) citește și formatează datele de intrare. Are formatul:

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

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

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

Astfel:

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

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

TEMA 1

Problema 1.1

Să se scrie un program care citește de la tastatură două numere întregi care reprezintă numărul de linii și numărul de coloane ale unei matrice cu elemente numere întregi. Programul va citi apoi matricea pe linii și o va afișa, de asemenea, pe linii.

Problema 1.2

Să se scrie o funcție care calculează transpusa unei matrice pătratice. Se vor afișa matricea inițială și cea transpusă.

Să se verifice relația:

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

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

Problema 1.3

Să se scrie un program care calculează suma și produsul a două matrice A și B. Elementele matricelor sunt numere întregi. Se vor scrie funcții pentru: citirea unei matrice, afișarea unei matrice pe monitor (în formatul cunoscut de la matematică – pe linii), calculul sumei a două matrice și calculul produsului a două matrice. Matricele nu sunt neapărat pătratice. Alegerea operației (suma sau produs de matrice) se va face de către utilizator la rularea programului. Pentru fiecare din cele două operații trebuie făcută validarea dimensiunilor matricelor astfel încât operația să poată fi făcută.

TEMA 2

Problema 2.1

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

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

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

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

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

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

Problema 2.2

Se citește o matrice A de dimensiune $n \times n$, cu $n \leq 20$ (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 funcții pentru: citire matrice, afișare matrice, interschimbare a două linii din matrice, interschimbare a două coloane dintr-o matrice, câte o funcție pentru fiecare opțiune de sortare. Cele două funcții de interschimbare vor primi ca parametri matricea, dimensiunea matricei, și cei doi indecși ai liniilor, respectiv coloanelor care se interschimbă.

TEMA 3

Problema 3.1

P3.1.a. Să se deducă relații de recurență pentru calcularea următoarelor matrice:

$$T_i = \frac{A^i}{i!}, \text{ cu } T_0 = I_n \text{ (unde } A \text{ este o matrice pătratică de ordin } n).$$

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

I_n este matricea unitate de dimensiune n .

P3.1.b. Să se scrie un program care

- 1). citește de la tastatură o matrice **A** de numere reale și de dimensiune **n** (cu **n** linii și **n** coloane), cu **n** ≤ 20 și afișează matricea citită.
- 2). Calculează matricea **B** cu expresia dată de relația (1)
- 3). Afișează 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 $n \leq 0$ sau $n \geq 20$ se va afișa un mesaj de eroare și se va cere o nouă valoare (corectă)).

Date de test:

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

și

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

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

Barem de notare:

1. Citire și validare dimensiune matrice	0,5
2. Citire matrice	1,0
3. Afișare matrice	1,0
4. Calcul matrice B	0,5
4a). Adunarea a două matrice	1,0
4b). Înmulțirea a două matrice	1,5
4c). Înmulțirea unei matrice cu un scalar	1,0
5. posibilitatea de reluare a programului	1,0
6. Apelul corect al funcțiilor	0,5
7. Acrierea corectă a funcției main	0,5
8. Folosirea relațiilor de recurență de terminate la punctul a).	1,5
TOTAL	10 p

Problema 3.2

Un teren de formă dreptunghiulară este împărțit 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 parcelor se memorează într-un tablou cu n linii și m coloane. Valoarea elementului de pe linia i și coloana j a tabloului este altitudinea celei de a j -a parcele de pe linia i .

Să se afișeze coordonatele “parceleur vârful”. O “parcelă vârful” are altitudinea strict mai mare decât a tuturor vecinilor săi.

Se va lua în considerare posibilitatea prelucrării mai multor seturi de date, la aceeași 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 afișeze matricea citită.

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

Să se determine imaginea în oglindă a matricii inițiale și să se afișeze.

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