FUNCȚII DE INTRARE – IEȘIRE CU FORMAT
Funcțiile scanf(...) și 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 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. Funcţia printf(...)

Funcţia printf(...) scrie, pe ecran, datele de ieșire aranjate (formatate). 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.

Formatarea datelor se face folosind specificatorii de format. Aceștia au următoarea formă generală (specificarea unui element între paranteze pătrate semnifică faptul că acel element este opțional):

%[flags][lăţime][.prec] tip_car

Fiecare specificator de format 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 câmpul de afișare cu spaţii);
  - **0n** vor fi afişate cel puţin n caractere, iar câmpul de afișare se completează cu zerouri plasate în fața valorii de afișat
- **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 (ca valoare și nu ca număr de cifre de afișat)

Dacă 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, D	Întreg	Întreg zecimal cu semn
o, O	Întreg	Întreg octal fără semn
u, U	Întreg	Întreg zecimal fără semn
x, X	Întreg	Întreg hexazecimal fără semn (cu litere mici)
f, 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

Exemple :

- 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 în 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 scanf(...)

Funcţia (cu prototipul tot în STDIO.H) citeşte de la tastatură şi formatează datele de intrare. Prototipul funcției este:

```
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** conţine specificatorii de format :

%[*][lăţime] tip_car

(specificarea unui element între paranteze pătrate semnifică faptul că acel element este opțional)

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
e	Real
f	Real
g	Real
o	Întreg octal
u	Întreg zecimal fără semn
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 ajung `n zonele de memorie rezervate 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.

Exemple:

- 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
- litera **x**: la fel cu **d**, dar se citeşte un întreg în forma hexazecimală
- litera **u**: citire întreg zecimal fără semn (**unsigned** - numere naturale)
- litera **f**: citire număr real simplă precizie (tip **float**)
- litera **e**: citire număr real (reprezentat în virgulă mobilă) în simplă precizie în forma cu exponent

- litera **g**: citire număr flotante în forma f sau e(E)
- litera **i**: poate însoţi d, o, x, u, f.
  ld, lo, lx: informația citită este memorată ca o data de tip long int
  lu: informația citită este memorată ca o dată unsigned long
  lf: informația citită este memorată ca o dată reprezentată în virgulă mobilă în dublă precizie (tipul double)
- litera **h**: poate însoţi d, o, x, u.
  hd, ho, hx: informația citită este memorată ca o data de tip short int
  hu: informația citită este memorată ca o dată unsigned short

- litera **c**: citire caracter curent chiar dacă este alb.

Exemplu3:

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

- litera **s**: citire tablou de caractere (șir de caractere) până la primul caracter alb sau până se atinge lungimea maximă. La sfârşitul unui şir citit astfel se pune automat '\0' – marcajul de sfârşit de şir.

Exemplu4:

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

Exemplu 5:

```
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'