

# Introducere în limbajul C (cont.)

9 octombrie 2002

- Câteva exemple
- Scheme logice și programe structurate
- Instrucțiuni de ciclare
- Tablouri

## Exemplu: Minimul a trei numere

---

```
#include <stdio.h>

void main(void)
{
    int a, b, c;

    printf("Introduceți trei numere întregi a b c: ");
    scanf("%d %d %d", &a, &b, &c); /* sau %d%d%d (la fel) */.
    printf("Cel mai mic număr este ");
    if (a < b) {
        if (a < c) printf("a = %d\n", a);
        else printf("c = %d\n", c);
    } else {
        if (b < c) printf("b = %d\n", b);
        else printf("c = %d\n", c);
    }
}
```

## Mai multe despre printf/scanf

---

În C, `printf/scanf` pot lua un număr arbitrar de argumente:

- primul este un șir de caractere (care indică formatul)
- restul: *expresii* (`printf`) sau *adrese* (`scanf`) cu tipuri corespunzătoare celor indicate în șirul de format
- pe rând, fiecare specificator de format din șir (de ex. `%d`, `%f`, etc.):
  - la `printf`: e înlocuit cu argumentul de pe poziția corespunzătoare din lista de argumente (o expresie)
  - la `scanf`: se citește o valoare de acel tip în adresa corespunzătoare din lista de argumente

## Exemplu: Ecuația de gradul II

---

```
#include <math.h>          /* funcții matematice */
#include <stdio.h>
void main(void)
{
    float a, b, c, b1, d, delta;
    printf("Coeficienții ecuației: "); /* ax^2+bx+c=0 */
    scanf("%f%f%f", &a, &b, &c);
    b1 = -b/(2.0*a); delta = b1*b1 - c/a;
    if (delta == 0) printf("Soluție dublă: %f\n", b1);
    else if (delta > 0) {
        d = sqrt(delta); /* rădăcină pătrată */
        printf("Soluții: %.1f și %.1f\n", b1+d, b1-d);
    } else /* delta < 0 */ {
        d = sqrt(-delta);
        printf("Soluții: %.1f+%.1fi și %.1f-%.1fi\n", b1, d, b1, d);
    }
}
```

## Ecuția de gradul II: comentarii

---

Coeficienții ecuației: 1 3 0

Soluții: 0.0 și -3.0

Coeficienții ecuației: 1 1 3

Soluții: -0.5+1.7i și -0.5-1.7i

- reamintim formula:  $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = -b/2a \pm \sqrt{(b/2a)^2 - c/a}$
- folosim tipul `float`: număr real (virgulă mobilă)
- formatarea (intrare și ieșire) cu `%f`
- opțional: specificarea preciziei, de ex. 2 cifre cu `%.2f`
- `sqrt` (radical): din biblioteca de funcții matematice standard
- specificatorii de formatare: încadrați în șirul de tipărit așa cum dorim să apară, fără a fi necesare spații. Ex. `%.1fi` ('i' e parte din text)

## Ciclul cu test inițial (instrucțiunea `while`)

---

`while ( expresie_logică )`

`instrucțiune`

`/* poate fi { compusă } */`

- se evaluează *expresia*
- dacă expresia e adevărată, se execută *instrucțiunea*, se reia totul de la evaluarea expresiei
- dacă expresia e falsă, ciclul se încheie (execuția continuă la punctul după *instrucțiune*)

Exemplu: calculați  $p = x^n$ , unde  $n \geq 0$

```
int n, p, x;
p = 1; /* valoarea initiala, x la puterea 0 */
while (n > 0) { /* n = de cate ori mai trebuie inmultit cu x */
    p = p * x; /* facem inmultirea */
    n = n - 1; /* a ramas cu o inmultire mai putin de facut */
}
```

## Ciclul cu test final (instrucțiunea `do ... while`)

---

```
do
    instrucțiune                /* poate fi { compusă } */
while ( expresie_logică );
```

- se execută *instrucțiunea*
- se evaluează *expresia*
- dacă expresia e *adevărată*, se revine la execuția *instrucțiunii*
- dacă expresia e *falsă*, ciclul se încheie

Exemplu:

```
char raspuns;
do {
    printf("Apăsați 'd' pentru a continua: ");
    scanf(" %c", &raspuns); /* citește caracter, sare peste spații */
} while (raspuns != 'd'); /* până la răspunsul dorit */
```

Obs: În Pascal, din `repeat ... until` se iese pe condiție *true* (invers!)

## Ciclul cu contor (instrucțiunea for)

```
for (exp_init ; exp_test ; exp_cont)
    instrucțiune
```

e echivalentă\* cu:

\* excepție: instrucțiunea `continue`, vezi ulterior

Calculul puterii:

```
for (p = 1; n > 0; n = n - 1)
    p = p * x;
```

- oricare din cele 3 expresii poate lipsi (dar cele două ; rămân)
- dacă `exp_test` lipsește, e tot timpul adevărată (ciclu infinit)

Cel mai simplu și frecvent caz: când știm numărul de iterații:

```
int i; /* variabilă cu care numărăm iterațiile */
for (i = 0; i <= 9; i = i + 1) /* 10 cicluri, de la 0 la 9 */
    printf("%d\n", i);
```

```
exp_init;
while (exp_test) {
    instrucțiune;
    exp_cont;
```

```
}
```

```
p = 1;
```

```
while (n > 0) {
```

```
    p = p * x;
```

```
    n = n - 1;
```

```
}
```

## Ciclul for (cont.)

---

Exemplu: calculul factorialului

```
int fact, i, n;
fact = 1;
for (i = 1; i <= n; i = i + 1) {
    fact = fact * i;
}
```

### Cicluri for imbricate

Exemplu: tiparirea numerelor de la 0 la 99:

```
int n, zeci, unit;
for (zeci = 0; zeci < 10; zeci = zeci + 1) {
    for (unit = 0; unit < 10; unit = unit + 1) {
        n = 10 * zeci + unit;
        printf("%d ", n); /* lasam un spatiu */
    }
    printf("\n"); /* dupa 10 numere, o linie nouă */
}
```

## Tablouri

---

- concept corespunzător șirului din matematică
- un șir de elemente de *același tip* de date
- un element: dat de *numele* tabloului, și un indice întreg: `x[3]`
- în C, numerotarea elementelor începe de la zero!!
- Sintaxa declarației: `tip nume_tablou[nr_elem];` (eventual inițializat)

Exemple: `int a[10];`      `float x[20];`

**Șiruri de caractere:** caz particular de tablouri

```
char cifre[20]; /* tablou de caractere neinițializat */
char nume[3] = { 'U', 'P', 'C' }; /* tablou inițializat */
char msg[] = "test"; /* constantă șir, se termină cu '\0' */
char msg[] = {'t','e','s','t','\0'}; /*același lucru scris altfel*/
```

- în memorie, sfârșitul unui șir e indicat cu caracterul special `'\0'` (nul)
- Atenție:** toate funcțiile care lucrează cu șiruri depind de acest lucru !  
(dar convenția nu are legătură cu aspectul în text, de ex. la citire)
- dacă șirul e declarat fără a explicita dimensiunea (vezi `msg`), se alocă dimensiunea inițializatorului (șirului dat) + 1 (pt. `'\0'`)

## Exemplu: cifrele unui număr în baza 10

---

```
/* conversie din număr în șir de cifre zecimale */
int cifre[20];          /* 20 de cifre e suficient */
int n, i = 0;          /* i e indicele în tablou */

scanf("%d", &n);        /* presupunem că n e dat corect */
do {
    cifre[i] = n % 10;  /* ultima cifra e restul împărțirii */
    i = i + 1;         /* avansăm pentru următoarea cifră */
    n = n / 10;        /* împărțire cu rest !! */
} while (n > 0);       /* i e numărul de cifre */
do {
    i = i - 1;         /* prima cifră e la poziția i - 1 */
    printf("%d", cifre[i]);
} while (i > 0);       /* ultima cifră e la poziția 0 */
```

## Tablouri multidimensionale

---

Există tablouri de oricâte dimensiuni. Pentru 2: matrici: `int m[10][7];`  
Interpretare: tablou de 10 elemente, fiecare un tablou de 7 întregi

```
void main(void) {
    int i, j, m[5][3], sumcol[3]; /* matrice și suma pe coloane */
    printf("Dați pe linii și coloane o matrice 5x3 de întregi\n");
    for (i = 0; i < 5; i++)
        for (j = 0; j < 3; j++)
            scanf("%d", &m[i][j]);
    for (j = 0; j < 3; j++) {
        sumcol[j] = 0;
        for (i = 0; i < 5; i++)
            sumcol[j] = sumcol[j] + m[i][j];
    }
}
```

**ATENȚIE** la depășirea limitelor tablourilor (eroare frecventă și gravă)!  
În limbajul C, verificarea cade în sarcina programatorului !