

CURS 04 – FP

Instrucțiunile limbajului C

4. instrucțiunea repetitivă cu număr fix de iterații - for

for(inițializări; condiții de continuare; acțiuni)
instrucțiune;

for(i = 0; i < 10; i++)
printf("%d ", i);

Variabila i controlează ciclul "for"!

Cea mai scurtă formă: for(; ;); => ciclare infinită deoarece condițiile de continuare se consideră implicit ca fiind adevărate!

Exemplu:

The screenshot shows a C code editor with the following code:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, j;
6
7     for(i = 1, j = 10; i < j; i++, j--)
8         printf("i = %d\tj = %d\n", i, j);
9
10    printf("\n");
11
12    return 0;
13 }
14
```

The output of the program is displayed in the console:

```
i = 1 j = 10
i = 2 j = 9
i = 3 j = 8
i = 4 j = 7
i = 5 j = 6
```

Process returned 0 (0x0) execution time : 0.020 s
Press any key to continue.

The screenshot shows the CodeBlocks IDE with a C program in the editor. The program includes `<stdio.h>` and defines a `main` function. Inside `main`, it declares two integer variables `i` and `j`. A `for` loop is used to iterate over values of `i` and `j`. The loop starts with `i = 1` and `j = 10`, and continues as long as `i < j`. Inside the loop, `printf` is used to print the current values of `i` and `j` in the format `"i = %d\tj = %d\n"`. After the loop, `printf` is used to print a newline character, and `return 0;` is used to end the program.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, j;
6
7     for(i = 1, j = 10; i < j;)
8     {
9         printf("i = %d\tj = %d\n", i, j);
10        i++;
11        j--;
12    }
13
14    printf("\n");
15
16    return 0;
17 }
```

The output window shows the execution results. The program prints the following output:

```
i = 1 j = 10
i = 2 j = 9
i = 3 j = 8
i = 4 j = 7
i = 5 j = 6
```

Process returned 0 (0x0) execution time : 0.015 s
Press any key to continue.

The screenshot shows the CodeBlocks IDE with a C program in the editor. The program includes `<stdio.h>` and defines a `main` function. Inside `main`, it declares two integer variables `i` and `j`. A `for` loop is used to iterate over values of `i` and `j`. The loop starts with `i = 1` and `j = 10`, and continues as long as `i < j`. Inside the loop, `printf` is used to print the current values of `i` and `j` in the format `"i = %d\tj = %d\n"`. After the loop, `printf` is used to print a newline character, and `return 0;` is used to end the program.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, j;
6
7     i = 1;
8     j = 10;
9     for(; i < j;)
10    {
11        printf("i = %d\tj = %d\n", i, j);
12        i++;
13        j--;
14    }
15
16    printf("\n");
17
18    return 0;
19 }
```

The output window shows the execution results. The program prints the following output:

```
i = 1 j = 10
i = 2 j = 9
i = 3 j = 8
i = 4 j = 7
i = 5 j = 6
```

Process returned 0 (0x0) execution time : 0.018 s
Press any key to continue.

Instrucțiunea for este o instrucțiune repetitivă cu test inițial!

Exemple:

1. Să se calculeze suma

$$S = 1^2 + 2^2 + \dots + n^2 = \sum_{i=1}^n i^2$$

```
#include <stdio.h>
int main()
{
    int i, s, n;

    printf("n = ");
    scanf("%d", &n);

    s = 0;
    for(i = 1; i <= n; i++)
        s = s + i*i;

    printf("Suma: %d\n", s);
    return 0;
}
```

Soluția eficientă: $S = \frac{n(n+1)(2n+1)}{6}$

2. Să se calculeze suma a n numere întregi citite de la tastatură.

```
#include <stdio.h>
int main()
{
    int i, s, n, x;

    printf("n = ");
    scanf("%d", &n);

    s = 0;
    // for(i = -1; i < n-1; i++)
    // for(i = 0; i < n; i++)
    for(i = 1; i <= n; i++)
    {
        printf("x = ");
        scanf("%d", &x);
        s = s + x;
    }
}
```

```

    printf("Suma: %d\n", s);
    return 0;
}

```

2. Să se calculeze suma cifrelor unui număr natural n citit de la tastatură.

$n \% 10$ = ultima cifră a numărului n

$n / 10$ = numărul obținut prin eliminarea ultimei cifre din numărul n

$5178 : 10 = 517$, rest 8

$n = 5178$	$s = 0$
$n = 517$	$s = 0 + 8 = 8$
$n = 51$	$s = 8 + 7 = 15$
$n = 5$	$s = 15 + 1 = 16$
$n = 0$	$s = 16 + 5 = 21$

Varianta 1 (cu instrucțiunea while):

```

#include <stdio.h>

int main()
{
    int s, n, aux;

    printf("n = ");
    scanf("%d", &n);

    //salvam valoarea lui n
    aux = n;

    //valoarea lui n va fi alterata (va deveni 0)
    s = 0;
    while(n != 0)
    {
        s = s + n%10;
        n = n / 10;
    }

    //restauram valoarea initiala a lui n
    n = aux;
}

```

```

        printf("Suma cifrelor lui %d: %d\n", n, s);

    return 0;
}

```

Varianta 2 (cu instrucțiunea for):

```

#include <stdio.h>

int main()
{
    int s, n, aux;

    printf("n = ");
    scanf("%d", &n);

    //salvam valoarea lui n
    aux = n;

    //valoarea lui n va fi alterata (va deveni 0)
    for(s = 0; n != 0; n = n / 10)
        s = s + n%10;

    //restauram valoarea initiala a lui n
    n = aux;

    printf("Suma cifrelor lui %d: %d\n", n, s);

    return 0;
}

```

Varianta 3 (cu un for ciudat)

```

#include <stdio.h>

int main()
{
    int s, n, aux;

    //initializari
    for(s = 0, printf("n = "), scanf("%d", &n), aux = n;
    //conditii de continuare
        n == 0 ? n = aux, printf("Suma cifrelor lui %d: %d\n", n, s), 0 : 1;
    //actiuni
        s = s + n%10, n = n / 10);

    return 0;
}

```

2. Să se afișeze suma a două numere naturale fără a folosi niciun operator.

```
#include<stdio.h>
int main()
{
    int n;

    //functia printf intoarce numarul caracterelor afisate pe ecran
    n = printf("Ana are mere!");

    //Numarul de caractere afisate: 13
    printf("\nNumarul de caractere afisate: %d\n", n);

    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int n;
    char x = 'A';

    //functia printf intoarce numarul caracterelor afisate pe ecran
    //caracterul 'A' va fi afisat pe 10 pozitii,
    //adica 9 spatii + 1 pozitie pentru 'A'
    n = printf("%10c", x);

    //Numarul de caractere afisate: 10
    printf("\nNumarul de caractere afisate: %d\n", n);

    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int n;
    char x = 'A';

    //functia printf intoarce numarul caracterelor afisate pe ecran
    //caracterul 'A' va fi afisat pe numarul de pozitii indicate de
    //prima valoare aflata dupa sirul de formatare, adica 30
    n = printf("%*c", 30, x);

    //Numarul de caractere afisate: 10
    printf("\nNumarul de caractere afisate: %d\n", n);

    return 0;
}
```

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int x, y;
```

```
    printf("x = ");
```

```
    scanf("%d", &x);
```

```
    printf("y = ");
```

```
    scanf("%d", &y);
```

```
    printf("\n%d + %d = %d\n", x, y, printf("%*c%c", x, ' ', y, ' '));
```

```
    return 0;
```

```
}
```

C:\Users\Ours\Desktop\Test_C\bin\Debug\Test_C.exe

x = 300

y = 567

300 + 567 = 867

Process returned 0 (0x0) execution time : 4.246 s

Press any key to continue.

Varianta finală:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int x, y;
```

```
    printf("x = ");
```

```
    scanf("%d", &x);
```

```
    printf("y = ");
```

```
    scanf("%d", &y);
```

```
    // '\r' = carriage return = intoarce cursorul la inceputul randului
```

```
    printf("%d + %d = %d", x, y, printf("%*c%c", x, '\r', y, '\r'));
```

```
    return 0;
```

```
}
```

4. Să se calculeze suma cifrelor fiecărui număr natural cuprins între două numere a și b citite de la tastatură.

a = 123

b = 200

Suma cifrelor numărului 123 este 6

Suma cifrelor numărului 124 este 7

.....

Suma cifrelor numărului 200 este 2

```

#include <stdio.h>

int main()
{
    int a, b, i, s, aux;

    printf("a = ");
    scanf("%d", &a);

    printf("b = ");
    scanf("%d", &b);

    //De obicei, NU trebuie sa alteram valoarea variabilei care
    //controleaza instructiunea for in interiorul sau!!!
    for(i = a; i <= b; i++)
    {
        //salvam valoarea variabilei care controleaza instructiunea for
        //intr-o variabila auxiliara si lucram cu ea
        aux = i;

        s = 0;
        while(aux != 0)
        {
            s = s + aux%10;
            aux = aux / 10;
        }

        printf("Suma cifrelor numarului %d este %d\n", i, s);
    }

    return 0;
}

```

5. instrucțiunea continue

Instrucțiunea continue întrerupe executarea iterației curente și forțează revenirea la începutul instrucțiunii repetitive (retestarea condiției).

Exemplu (afișarea numerelor impare <= n):

```

#include <stdio.h>

int main()
{
    int i, n = 10;

    for(i = 0; i < n; i++)
    {
        if(i % 2 == 0)
            continue;

        printf("%d ", i);
    }
}

```



```

    }

    return 0;
}

```

Exemplu (afișarea numerelor $\leq n$, mai puțin 7 și 9):

```

#include <stdio.h>

int main()
{
    int i, n = 10;

    for(i = 0; i < n; i++)
    {
        if(i == 7 || i == 9)
            continue;

        printf("%d ", i);
    }

    return 0;
}

```

5. instrucțiunea break

Instrucțiunea **break** întrerupe executarea întregii instrucțiuni repetitive și forțează trecerea la următoarea instrucțiune din program.

Exemplu (suma numerelor dintr-un șir de numere întregi terminat cu 0):

12, -3, 7, 5, -10, 8, 0 => suma = 19

```

#include <stdio.h>

int main()
{
    int x, s;

    s = 0;
    while(1)
    {
        printf("x = ");
        scanf("%d", &x);
    }
}

```

```

        if(x == 0)
            break;

        s = s + x;
    }

    printf("Suma: %d", s);

    return 0;
}

```

Exemplu (testarea primalității unui număr):

Un număr este *prim* dacă se divide doar cu 1 și el însuși.

Un număr este *prim* dacă nu are divizori proprii (cuprinși între 2 și jumătatea sa).

Varianta 1:

```

#include <stdio.h>

int main()
{
    int n, d, prim;

    printf("n = ");
    scanf("%d", &n);

    prim = 1;
    for(d = 2; d <= n/2; d++)
        if(n % d == 0)
            prim = 0;

    if(prim == 1)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);

    return 0;
}

```

Varianta 2 (se oprește testarea la primul divizor propriu găsit):

```

#include <stdio.h>

int main()
{
    int n, d, prim;

```

```

printf("n = ");
scanf("%d", &n);

prim = 1;
for(d = 2; d <= n/2 && prim == 1; d++)
    if(n % d == 0)
        prim = 0;

if(prim == 1)
    printf("Numarul %d este prim!", n);
else
    printf("Numarul %d este compus!", n);

return 0;
}

```

Varianta 3 (cu instrucțiunea break)

```

#include <stdio.h>

int main()
{
    int n, d;

    printf("n = ");
    scanf("%d", &n);

    for(d = 2; d <= n/2; d++)
        if(n % d == 0)
            break;

    if(d == n/2 + 1)
        printf("Numarul %d este prim!", n);
    else
        printf("Numarul %d este compus!", n);

    return 0;
}

```

Teoremă:

Dacă un număr natural $n \geq 2$ este compus, atunci el are cel puțin un divizor $d \leq \sqrt{n}$.

Demonstrația: Presupunem prin absurd faptul că numărul $n \geq 2$ este compus, dar nu are niciun divizor $d \leq \sqrt{n} \Rightarrow$ numărul n are cel puțin 2 divizori proprii

$d_1 > \sqrt{n}$ și $d_2 > \sqrt{n}$ astfel încât $d_1 \cdot d_2 = n$. Dar $d_1 \cdot d_2 > \sqrt{n} \cdot \sqrt{n} \Rightarrow n > n$ (contradicție!), deci presupunerea făcută este falsă!

```
#include<stdio.h>

int main()
{
    int divizor, n;

    printf("n = ");
    scanf("%d", &n);

    //divizor * divizor <= n este echivalenta cu
    //divizor <= sqrt(n), dar este mai rapida
    for(divizor = 2; divizor * divizor <= n; divizor++)
        if(n % divizor == 0)
            break;

    if(divizor == n/2 + 1)
        printf("Numarul %d este prim!\n", n);
    else
        printf("Numarul %d este compus!\n", n);

    return 0;
}
```