INFORMATICĂ · CLASA A IX-A · STRUCTURI DE CONTROL · LECȚIA 6

Structuri repetitive (2): for, break, continue și bucle imbricate

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-05

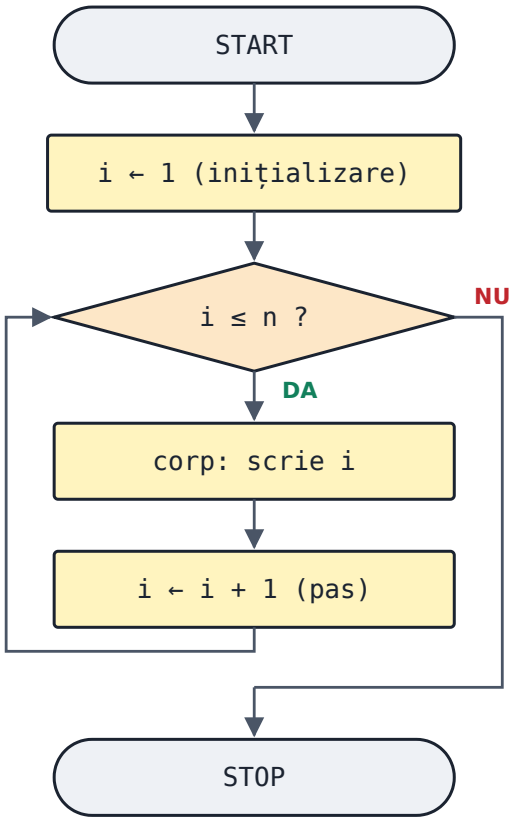
CE VEI ȘTI DUPĂ LECȚIE

- Folosești for când știi dinainte de câte ori se repetă (citirea a n numere, parcurgerea 1..n).
- Numeri corect de câte ori se execută o buclă (exercițiu frecvent la Bac).
- Folosești break și continue fără să complici programul.
- Scrii bucle imbricate (o buclă în altă buclă) și desenezi figuri din caractere.

1. Când folosim `for`?

La lecția precedentă ai repetat cu `while` (nu știi de câte ori) și `do-while`. Foarte des **știi dinainte** de câte ori: „citește n numere", „afișează de la 1 la n", „parcurge fiecare cifră". Pentru aceste cazuri, `for` strânge într-un singur loc cele trei lucruri care la `while` sunt împrăștiate: de unde pornesc, până când merg și cum avansez.

```
for (initializare; conditie; pas)
{
    // corp
}
```



ea se face o singură dată; condiția se testează înaintea fiecă

Ce face for

Ordinea exactă de execuție:

1. **inițializare** (o singură dată): `i = 1`;
2. se verifică **condiția** `i <= n`; dacă e falsă, bucla se termină;
3. se execută **corpul**;
4. se execută **pasul** `i++`;
5. înapoi la pasul 2.

Așadar `for` este doar o scriere compactă pentru:

```
int i = 1;
while (i <= n)
{
    cout << i << ' ';
    i++;
}
```

Urmărire pentru `n = 3`:

pas	i	`i <= 3` ?	corp (afișare)	după pas
1	1	da	`1`	i = 2
2	2	da	`2`	i = 3

pas	i	`i <= 3` ?	corp (afișare)	după pas
3	3	da	`3`	i = 4
4	4	**nu** → ies		

Observă: condiția se testează **o dată în plus** decât numărul de iterații, iar la ieșire `i = n + 1`.

DE CE AȘA?

De ce `for` și nu mereu `while`? Cele trei părți la un loc previn cea mai frecventă greșeală din lecția trecută: uitarea pasului. Cine citește `for (i = 1; i <= n; i++)` vede imediat „de la 1 la n, din 1 în 1". Alege `while` când nu știi numărul de pași (citire până la 0, algoritmul lui Euclid), `for` când îl știi.

Pseudocod ↔ C++

```
┌pentru i ← 1, n execută        for (int i = 1; i <= n; i++)
│   scrie i                         cout << i;
└■
┌pentru i ← n, 1, -1 execută    for (int i = n; i >= 1; i--)
└■
┌pentru i ← 2, n, 2 execută     for (int i = 2; i <= n; i += 2)
└■
```

LA BAC

Întrebare tipică la BAC: „De câte ori se execută corpul?" Pentru `pentru i ← a, b execută` (pas 1) răspunsul este **b − a + 1** (dacă a ≤ b). Pentru pas k: ⌊(b − a)/k⌋ + 1. Verifică: `i = 1, 4, 7, 10` → (10 − 1)/3 + 1 = 4.

2. Citirea a n numere

Spre deosebire de „până la 0", aici știm câte numere sunt:

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n, x, s = 0;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> x;
        s += x;
    }
    cout << s;
    return 0;
}
```

intrare	n	după citire i=1	i=2	i=3	ieșire
`3` / `5 2 8`	3	s = 5	s = 7	s = 15	`15`

DE CE AȘA?

De ce se declară `x` în afara buclei, dar `i` în `for`? Variabila `i` e doar contorul și nu mai e nevoie de ea după buclă. `x` și `s` trebuie să trăiască după (s e afișat la final), deci se declară înainte. Regula: o variabilă există în blocul în care a fost declarată.

SFAT

Dacă ai nevoie de valoarea lui `i` după buclă, declar-o înainte: `int i; for (i = 1; ...)`.

3. `break` și `continue`

- `break`: iese **imediat** din bucla curentă;
- `continue`: sare la **următoarea** iterație (în `for`, se execută pasul).

```
for (int i = 1; i <= 6; i++)
{
    if (i % 2 == 0) continue;   // sare peste numerele pare
    if (i == 5) break;          // oprește bucla când ajunge la 5
    cout << i << ' ';
}
```

i	`i % 2 == 0`	`i == 5`	se afișează
1	nu	nu	`1`
2	**da** → continue		
3	nu	nu	`3`
4	**da** → continue		
5	nu	**da** → break	(nimic, ies)

Se afișează `1 3`. Exemplu util: primul număr par dintre n citite:

```
int n, x, primPar = -1;
cin >> n;
for (int i = 1; i <= n; i++)
{
    cin >> x;
    if (x % 2 == 0) { primPar = x; break; }
}
```

ATENȚIE — GREȘELI FRECVENTE

Dacă ieși cu `break` dintr-o buclă care citește date, restul numerelor rămân necitite. Pe pbinfo nu contează (programul se termină), dar dacă după buclă mai urmează alte citiri, vor citi valori greșite. În acest caz folosește o variabilă `gasit` în loc de `break`.

DE CE AȘA?

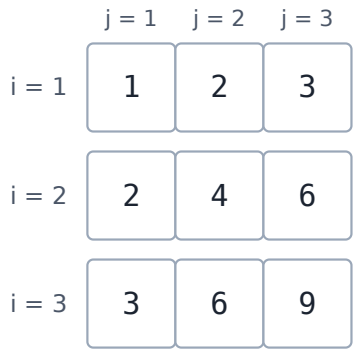
De ce există `break` dacă putem folosi condiția buclei? Pentru a opri imediat, fără să mai executăm restul corpului; ideal la căutări („am găsit, nu mai caut"). Dar un `break` ascuns într-o buclă lungă o face greu de citit: folosește-l rar și clar.

4. Bucle imbricate

O buclă în corpul altei bucle. Bucla interioară se execută **complet** pentru fiecare pas al celei exterioare, ca minutarul față de ora de pe ceas: pentru fiecare oră, minutarul face un tur întreg.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
            cout << i * j << ' ';
        cout << '\n';
    }
    return 0;
}
```



are i, bucla interioară parcurge o lin

Tabla înmulțirii

Corpul interior se execută de `n · n` ori. Urmărire pentru `n = 3`, primele rânduri:

i	j	afișează
1	1, 2, 3	`1 2 3` apoi rând nou
2	1, 2, 3	`2 4 6` apoi rând nou
3	1, 2, 3	`3 6 9`

DE CE AȘA?

De ce `cout << '\n'` în bucla exterioară, nu în cea interioară? Rândul nou se pune **după ce s-a terminat un rând întreg**, adică după bucla interioară, dar în corpul celei exterioare. Pus înăuntru, ai afișa fiecare număr pe rând separat.

Desene din caractere

Ideea: bucla exterioară = **rândurile**, bucla interioară = **ce afișăm pe rândul i**.

	1	2	3	4	5
1	*				
2	*	*			
3	*	*	*		
4	*	*	*	*	
5	*	*	*	*	*

(bucla exterioară), coloanele 1..i (bucla int

Triunghi de steluțe

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= i; j++)    // pe rândul i sunt i steluțe
            cout << '*';
        cout << '\n';
    }
    return 0;
}
```

Intrare / ieșire:

Intrare	Ieșire
`4`	`*` / `**` / `***` / `****` (pe rânduri separate)

Pentru piramida `1`, `1 2`, `1 2 3` ... (`#351`), pe rândul `i` afișezi numerele de la 1 la `i`.

SFAT

Pentru orice desen: scrie pe hârtie, pentru fiecare rând `i`, **câte** caractere de fiecare tip apar (spații, steluțe) ca formulă de `i` și `n`. Apoi transformi fiecare formulă într-o buclă. De exemplu, o piramidă cu vârful sus are pe rândul `i`: `n - i` spații și `2*i - 1` steluțe.

5. Câte operații face o buclă imbricată?

Secvență	Câte execuții ale corpului
`for i=1..n`	n
`for i=1..n` / `for j=1..n`	n^2
`for i=1..n` / `for j=1..i`	1 + 2 + … + n = n(n+1)/2
`for i=1..n` / `for j=i+1..n`	n(n−1)/2 (perechi distincte)

Verificare pentru `n = 4` la a treia: 1 + 2 + 3 + 4 = 10 = 4 · 5 / 2.

DE CE AȘA?

De ce contează numărul de execuții? El îți spune cât de lent este programul. O buclă simplă cu n = 10^5 face 10^5 pași (instant), dar două bucle imbricate fac 10^{10} (zeci de secunde). Așa înveți să estimezi dacă o soluție trece de limita de timp, tema lecției 14.

6. Greșeli frecvente

Greșeala	Efect	Corect
`for (i = 1; i <= n; i++);`	`;` face corpul gol; `cout` rulează o singură dată	fără `;`
`i < n` în loc de `i <= n`	se pierde ultimul element	verifică la capete
același contor în ambele bucle imbricate	bucla exterioară se strică	`i` și `j` diferite
modifici `i` în corp	pași sărite, uneori buclă infinită	nu modifica contorul
`cout << '\n'` pus în buclă greșită	rânduri greșite	după bucla interioară

7. La BAC

LA BAC

Tipare întâlnite des: (1) „de câte ori se afișează…" (formula b − a + 1 sau suma 1 + 2 + …), (2) „ce afișează algoritmul pentru n = …" (tabel cu `i`, `j`), (3) „înlocuiți `pentru` cu `cât timp`". La (3): `i ← a` înainte, `cât timp i ≤ b`, pasul la sfârșitul corpului.

8. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Ce afișează pentru `n = 4`?

```
citește n
┌pentru i ← 1, n execută
│   ┌pentru j ← i, n execută
│   │   scrie j
│   └■
└■
```

2. De câte ori se afișează `*`? `for (i = 1; i <= 10; i += 3) cout << '*';`

3. Rescrie bucla de la 2. folosind `while`.

4. Ce afișează `for (int i = 5; i >= 1; i--) cout << i;`?

5. Ce afișează?

```cpp
for (int i = 1; i <= 6; i++)
{
    if (i % 3 == 0) continue;
    if (i == 5) break;
    cout << i;
}
```

6. De câte ori se execută corpul pentru `pentru i ← 3, 17, 2`?

7. Ce valoare are `s` după secvență?

```cpp
int s = 0;
for (int i = 1; i <= 4; i++)
    for (int j = 1; j <= i; j++)
        s++;
```

8. Ce afișează `s` la final și de ce?

```cpp
int s = 0;
for (int i = 1; i <= 4; i++);
    s += i;
```

(Presupunem că `i` a fost declarat înainte.)

9. Scrie bucla care afișează pătratele perfecte mai mici sau egale cu `n`, pe un rând.

10. Pentru `n = 3`, ce desen produce?

```cpp
for (int i = n; i >= 1; i--)
{
    for (int j = 1; j <= i; j++) cout << '#';
    cout << '\n';
}
```

9. Mai departe (pentru cine vrea mai mult)

Suma lui Gauss. 1 + 2 + … + n = n(n+1)/2: o buclă de un miliard de pași devine o singură formulă. Legenda spune că Gauss a descoperit-o la 9 ani, pe bancă. Verifică formula cu o buclă pentru n = 100: 5050.

Bucle în bucle în bucle. Trei bucle imbricate cu `n` pași fac n^3. Pentru n = 1000 sunt un miliard de pași: aproape de limita unei secunde. Așa afli de ce un algoritm „corect" poate totuși să pice pe pbinfo cu „timp depășit".

Mai multe variabile în `for`. `for (int i = 1, j = n; i < j; i++, j--)` are două contoare care se apropie unul de altul. Este exact tiparul de comparare a cifrelor/elementelor de la capete spre mijloc (palindrom, oglindit).

Range-for (C++11). Pentru vectori vei putea scrie `for (int x : v)`: „pentru fiecare x din v". Vei folosi forma asta în lecțiile cu vectori și în orice cod modern.

Rezumat

- `for (init; cond; pas)`: se folosește când știi numărul de pași.
- Numărul de execuții: b − a + 1 (pas 1).
- `break` iese din buclă; `continue` trece la pasul următor.
- Bucle imbricate: exterioara = rânduri, interioara = coloane; corpul interior se execută de n·n ori.
- Pentru desene: formulă pe rând, apoi o buclă pentru fiecare tip de caracter.

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] **#327 afisarenumere** — ușoară
  Primele n numere naturale nenule, crescător.
  https://www.pbinfo.ro/probleme/327/afisarenumere
- [ ] **#2 suma-n** — ușoară
  Se citește n, apoi n numere: suma lor.
  https://www.pbinfo.ro/probleme/2/suma-n
- [ ] **#348 afisareputeri** — ușoară
  Puterile lui n mai mici sau egale cu p — atenție la depășire.
  https://www.pbinfo.ro/probleme/348/afisareputeri
- [ ] **#351 piramida** — ușoară
  Prima buclă imbricată.
  https://www.pbinfo.ro/probleme/351/piramida

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #328 **afisarenumere1** — ușoară
  https://www.pbinfo.ro/probleme/328/afisarenumere1
- [ ] #330 **afisarenumerepare** — ușoară
  https://www.pbinfo.ro/probleme/330/afisarenumerepare
- [ ] #331 **afisarenumereimpare** — ușoară
  https://www.pbinfo.ro/probleme/331/afisarenumereimpare
- [ ] #339 **numere** — ușoară
  Primul număr par dintre cele n: break sau o variabilă de tip „găsit".
  https://www.pbinfo.ro/probleme/339/numere
- [ ] #326 **ultimulpar** — ușoară
  https://www.pbinfo.ro/probleme/326/ultimulpar
- [ ] #115 **medie** — ușoară
  Media cu zecimale: atenție la împărțire.
  https://www.pbinfo.ro/probleme/115/medie
- [ ] #350 **produscartezian** — ușoară
  https://www.pbinfo.ro/probleme/350/produscartezian
- [ ] #457 **patrat** — ușoară
  https://www.pbinfo.ro/probleme/457/patrat

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #1376 **laturitriunghi** — ușoară
  https://www.pbinfo.ro/probleme/1376/laturitriunghi
- [ ] #2530 **desenclepsidra** — ușoară
  https://www.pbinfo.ro/probleme/2530/desenclepsidra

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #796 **Exercițiul #796**
  pseudocod cu pentru: ce se afișează (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=796
- [ ] #797 **Exercițiul #797**
  pseudocod cu pentru, variantă (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=797

- [ ] #815 **Exercițiul #815**
  bucle imbricate: câte afișări (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=815

- [ ] #817 **Exercițiul #817**
  bucle imbricate, variantă (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=817

- [ ] #977 **Exercițiul #977**
  trei secvențe S1–S3: care fac același lucru (Bac 2018)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=977

- [ ] #1244 **Exercițiul #1244**
  bucle imbricate cu i<4 și j<5 (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1244

- [ ] #1210 **Exercițiul #1210**
  bucla cu un caracter c (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1210

- [ ] #1405 **Exercițiul #1405**
  pseudocod cu pentru i←1,n (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1405

- [ ] #1406 **Exercițiul #1406**
  pseudocod cu pentru i←1,n, variantă (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1406

- [ ] #1693 **Exercițiul #1693**
  expresia care completează o buclă (Bac 2025)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1693

- [ ] #1714 **Exercițiul #1714**
  expresia care completează o buclă cu n≥2 (Bac 2025)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1714

Citește și exersează în plus

- Instrucțiunea for https://www.pbinfo.ro/articole/26011/instructiunea-for
- Instrucțiunea break https://www.pbinfo.ro/articole/26012/instructiunea-break
- Instrucțiunea continue https://www.pbinfo.ro/articole/26013/instructiunea-continue
- Python Tutor pentru C++ — urmărește pas cu pas ordinea: inițializare, condiție, corp, pas https://pythontutor.com/cpp.html
- cppreference — bucla for — referința oficială (engleză) https://en.cppreference.com/w/cpp/language/for
- Programarea ca joc: Code.org / Hour of Code — bucle explicate vizual, bun pentru a le arăta cuiva mai mic https://hourofcode.com/ro

Răspunsuri

Exersează pe hârtie

1. `1234234344`. Pentru i = 1: 1 2 3 4; i = 2: 2 3 4; i = 3: 3 4; i = 4: 4 (fără spații între ele).

2. 4 ori (i = 1, 4, 7, 10).

3.

```
int i = 1;
while (i <= 10)
{
    cout << '*';
    i += 3;
}
```

4. `54321`.

5. `124`. i = 1: afișează 1; i = 2: afișează 2; i = 3: continue (nu afișează); i = 4: afișează 4; i = 5: break.

6. 8 ori (i = 3, 5, 7, 9, 11, 13, 15, 17). Formula: (17 − 3)/2 + 1 = 8.

7. `10` (1 + 2 + 3 + 4).

8. `s += i` se execută **o singură dată**, după buclă, cu i = 5 (valoarea de după ieșire), deci s = 5. Cauza este `;` după `for`, care face corpul buclei gol.

9.

```
for (int i = 1; i * i <= n; i++)
    cout << i * i << ' ';
```

10. Triunghi inversat:

```
###
##
#
```