INFORMATICĂ · CLASA A VI-A · STRUCTURI REPETITIVE · LECȚIA 3

# Structura repetitivă condiționată anterior (cât timp ... execută)

Durată: 45 min  Nivel: gimnaziu  Înainte: lecțiile g6-02

**CE VEI ȘTI DUPĂ LECȚIE**

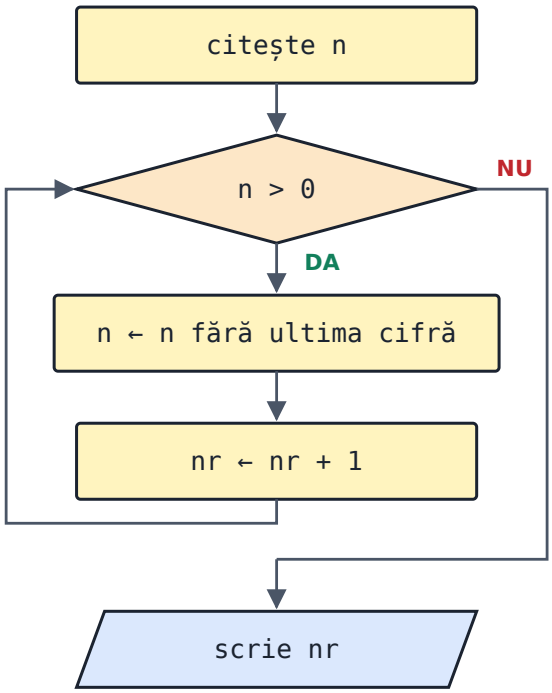
- Folosești repetiția când nu știi dinainte de câte ori se repetă.
- Înțelegi că testul se face ÎNAINTE de fiecare repetare.
- Folosești în Scratch blocul „repetă până când” și înțelegi legătura cu „cât timp”.
- Urmărești un algoritm cu „cât timp” într-un tabel.

## 1. Când nu știm de câte ori

„Mergi înainte **cât timp** nu ai ajuns la perete.” Nu știm câți pași sunt: depinde de cameră. Repetăm **cât timp** o condiție este adevărată și ne oprim când devine falsă.

```
cât timp nu am ajuns la perete execută
    fă un pas
sfârșit cât timp
```

Condiția se verifică **înainte** de fiecare pas. Dacă de la început ești lângă perete, nu faci niciun pas (**zero repetări** este posibil).




testul „n > 0” se face ÎNAINTE de fiecare repetare

*Schema logică*

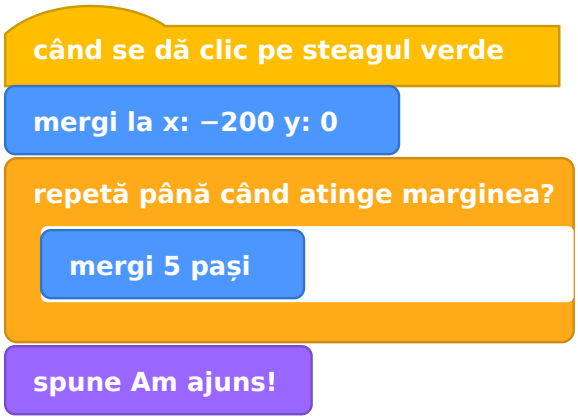
**DE CE AȘA?**

**De ce testul este în romb, înaintea corpului?** Pentru că înainte de orice pas ne întrebăm „mai merită să continui?”. Dacă răspunsul este NU de la început, săgeata NU ocolește corpul și nu se execută nimic.

## 2. În Scratch: „repetă până când”

Scratch are blocul `repetă până când < condiție >`. El verifică tot **înainte**, dar se oprește când condiția devine **adevărată**. Deci:

> „repetă până când **ating marginea**” = „cât timp **NU** ating marginea, repetă”




*Scratch*

**DE ȘTIUT**

Același algoritm se poate scrie cu „cât timp condiția" sau cu „până când opusul condiției". Important este să știi **când se oprește**: la „cât timp" se oprește când condiția devine **falsă**; la „repetă până când" se oprește când devine **adevărată**.

## 3. Exemplu numeric: câte cifre are un număr?

Două operații utile: `n mod 10` = ultima cifră (rest la împărțirea cu 10) și `n div 10` = n **fără** ultima cifră (câtul împărțirii; 365 div 10 = 36).

```
citește n
nr ← 0
cât timp n > 0 execută
    n ← n div 10
    nr ← nr + 1
sfârșit cât timp
scrie nr
```

Urmărirea pentru n = 365:




când testul devine fals, repetiția se oprește și afișăm nr = 3

*Urmărire*

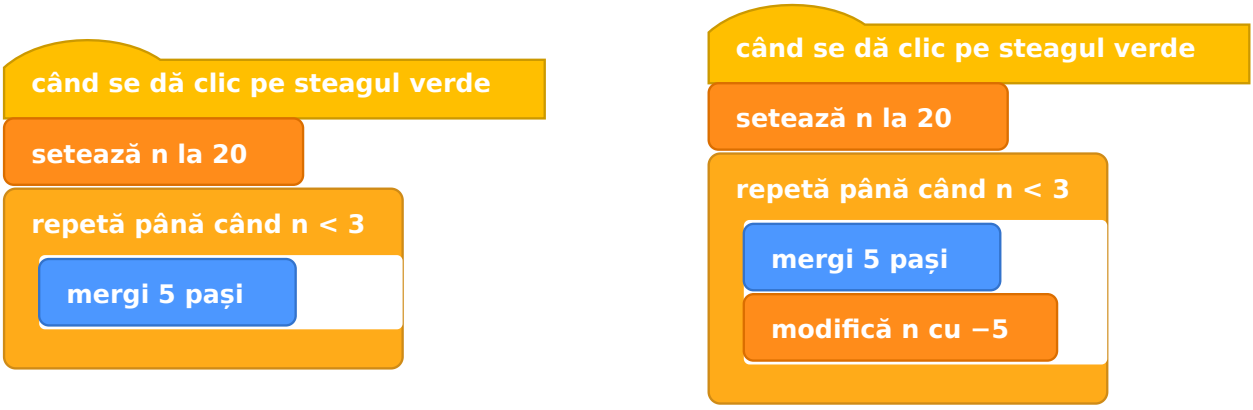
| Pas | n înainte | n > 0? | n după | nr |
|---|---|---|---|---|
| 1 | 365 | da | 36 | 1 |
| 2 | 36 | da | 3 | 2 |
| 3 | 3 | da | 0 | 3 |
| 4 | 0 | **nu** → stop | | **3** |

**DE CE AȘA?**

**De ce se oprește?** La fiecare pas n scade (se pierde o cifră) și în cele din urmă ajunge 0, când condiția `n > 0` devine falsă. Orice repetiție trebuie să facă, în interior, ceva care **apropie condiția de oprire**.

## 4. Atenție: bucla care nu se mai oprește

Dacă în interior nu se schimbă nimic din condiție, repetiția nu se termină niciodată („**buclă infinită**"). În Scratch apeși butonul roșu de stop.




*Buclă infinită*

## 5. Greșeli frecvente

| Greșeala | Efect | Corect |
|---|---|---|
| uiți să schimbi variabila din condiție | buclă infinită | `n ← n div 10`, `modifică n cu −5` |
| condiție greșită (`n ≥ 0` în loc de `n > 0`) | un pas în plus; la cifre numără 4 în loc de 3 | verifică cu un exemplu |
| confunzi „cât timp” cu „până când” | opusul condiției | cât timp C = până când nu C |
| uiți inițializarea (`nr ← 0`) | rezultate ciudate | pune valoarea de început |
| pui prea multă pauză în interior | joc lent | `așteaptă` mic |

## 6. Exersează pe hârtie

**1.** Urmărește în tabel: `n ← 20`, `cât timp n > 3 execută n ← n − 5`. Ce valoare are n la final? De câte ori s-a repetat?

**2.** Câte cifre are 7? Dar 1000? Aplică algoritmul din lecție.

**3.** Suma cifrelor lui 253: `s ← 0`, `cât timp n > 0 execută s ← s + n mod 10, n ← n div 10`. Completează tabelul.

**4.** Câte repetări are `x ← 1; cât timp x < 50 execută x ← x * 2`? Care este valoarea finală a lui x?

**5.** Câte repetări are `cât timp 1 = 2 execută …`? Dar `cât timp 1 = 1 execută …` (fără nicio schimbare)?

**6.** Scrie condiția pentru „cât timp pisica nu atinge peretele” folosind `repetă până când` în Scratch.

**7.** Algoritmul „afișează numerele de la 1 la 5” cu `cât timp`: completează `i ← 1`, `cât timp ___ execută`, `scrie i`, `___`, `sfârșit cât timp`.

**8.** Ce afișează `n ← 12`, `cât timp n > 0 execută scrie n, n ← n − 4`?

**9.** Care este ultima cifră a lui 4079? Dar numărul fără ultima cifră? (cu `mod` și `div`)

**10.** De ce poate „cât timp" să nu se execute deloc, iar „pentru i ← 1, n" (cu n ≥ 1) se execută mereu cel puțin o dată?

## 7. Provocări în Scratch

1. Personajul cade de sus (y scade cu 3) **până când** atinge pământul (o culoare verde desenată jos).
2. Joc „ghicește numărul": calculatorul alege un număr între 1 și 20; **repetă până când** răspunsul e corect, spunând „mai mare" sau „mai mic" (deocamdată cu `întreabă`).
3. Pisica merge la dreapta **până când** atinge marginea, apoi se întoarce și merge înapoi.
4. **Dificil:** un program care numără câte cifre are numărul introdus (algoritmul din lecție, în Scratch cu `restul împărțirii` și `rotunjește în jos`).

## 8. Mini-grile

**1.** La `cât timp C execută`, corpul se execută de cel puțin: a) 0 ori b) 1 dată c) 2 ori d) de n ori

**2.** `repetă până când < atinge marginea? >` se oprește când: a) atinge marginea b) nu atinge marginea c) după 10 pași d) niciodată

**3.** Pentru n = 365, câte repetări face algoritmul de numărare a cifrelor? a) 2 b) 3 c) 4 d) 365

**4.** Cum se numește repetiția a cărei condiție nu devine niciodată falsă? a) buclă infinită b) buclă scurtă c) buclă pentru d) buclă goală

## 9. Mai departe (pentru curioși)

**`mod` și `div` în programare.** Aceste operații sunt peste tot: `mod` pentru „par/impar", „multiplu de", ultima cifră; `div` pentru a „tăia" cifre. În C++ se scriu `%` și `/` (între numere întregi), în Python `%` și `//`.

**De la cifre la numere speciale.** Cu `cât timp` + `mod` + `div` poți verifica: este un număr palindrom (121)? are toate cifrele pare? care e cea mai mare cifră? Sunt problemele clasice din clasa a IX-a.

**Viteza.** Un calculator face miliarde de pași pe secundă: o buclă infinită „îngheață" programul imediat. De aceea programatorii gândesc mereu: „ce se schimbă în interior ca să ajung la oprire?".

## 10. Ce am învățat

- `cât timp condiție` = repetă cât e adevărată; testul e înainte (se poate să nu repete deloc).
- În Scratch: `repetă până când` se oprește când condiția devine adevărată.
- În interior trebuie să se schimbe ceva, altfel bucla nu se oprește.
- `n mod 10` = ultima cifră; `n div 10` = n fără ultima cifră.

## Citește și exersează în plus

- Scratch — editorul online — aici rulezi exemplele din lecție https://scratch.mit.edu/projects/editor/

- Scratch Wiki — Control blocks — repetările în Scratch: repetă, la nesfârșit, repetă până când (engleză) https://en.scratch-wiki.info/wiki/Control_Blocks
- Code.org — Hour of Code — puzzle-uri cu repetiții și condiții, gratuite https://code.org/hourofcode/overview

# Răspunsuri

## Exersează pe hârtie

**1.**

| | n | n > 3? |
|---|---|---|
| început | 20 | |
| test n > 3 | 20 | **da** |
| n ← n − 5 | 15 | |
| test | 15 | **da** |
| n ← n − 5 | 10 | |
| test | 10 | **da** |
| n ← n − 5 | 5 | |
| test | 5 | **da** |
| n ← n − 5 | 0 | |
| test | 0 | **nu → stop** |

repetat de 4 ori; la final n = 0

*Urmărire n = 20*

Se repetă de **4 ori**; la final **n = 0**.

**2.** 7 are **1** cifră (7 → 0 într-un pas); 1000 are **4** cifre (1000 → 100 → 10 → 1 → 0).

**3.** n = 253: s = 3, n = 25; s = 3 + 5 = 8, n = 2; s = 8 + 2 = 10, n = 0. Suma cifrelor: **10**.

**4.** x = 1, 2, 4, 8, 16, 32, 64: la 64 testul `x < 50` este fals. S-a repetat de **6** ori; x final = **64**.

**5.** `cât timp 1 = 2` nu se execută deloc (**0** repetări); `cât timp 1 = 1` fără modificări se repetă **la infinit**.

**6.** `repetă până când < atinge peretele? >` (se oprește când atinge peretele).

**7.** `i ← 1` / `cât timp i ≤ 5 execută` / `scrie i` / `i ← i + 1` / `sfârșit cât timp`.

**8.** n = 12, 8, 4, apoi 0 (oprire). Afișează **12 8 4**.

**9.** 4079 mod 10 = **9**; 4079 div 10 = **407**.

**10.** La „cât timp" condiția se verifică la început și poate fi falsă de la început → 0 repetări. La „pentru i ← 1, n" cu n ≥ 1 sunt n repetări, deci cel puțin una.

## Mini-grile

**1.** a. **2.** a. **3.** b. **4.** a.

---