INFORMATICĂ · CLASA A IX-A · ALGORITMI · LECȚIA 7

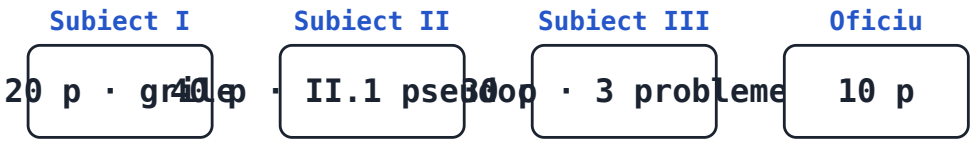
Pseudocod ↔ C++: exercițiul de pseudocod de la Bac

Durată: 60 min Nivel: mate-info (Bac) Înainte: lecțiile ix-06

CE VEI ȘTI DUPĂ LECȚIE

- Cunoști structura probei de informatică de la Bac și unde apare pseudocodul.
- Rezolvi metodic itemii a), b), c), d) ai exercițiului II.1.
- Traduci corect orice structură din pseudocod în C++ și invers.
- Găsești valori de intrare pentru care algoritmul afișează un rezultat dat.

1. Cum arată proba de informatică (mate-info, C++)



II.1 se rezolvă integral cu materia de clasa a IX-a

Structura probei

Subiect	Punctaj	Ce conține
I	20 p	5 întrebări grilă (expresii, structuri, recursivitate, grafuri, backtracking)
II	40 p	**II.1 – algoritm în pseudocod** (a, b, c, d) + două itemi scurți (șiruri de caractere, structuri, matrice)
III	30 p	trei probleme: un subprogram, un program, o problemă cu fișier + algoritm eficient
oficiu	10 p	

Exercițiul II.1 se poate rezolva **complet** cu materia de clasa a IX-a și aduce mulți dintre cei 40 de puncte. E cel mai „sigur” exercițiu din toată proba: trebuie să îl iei integral.

DE CE AȘA?

De ce îl învățăm acum, în clasa a IX-a? Pentru că pseudocodul este *aceeași* limbă de gândire ca schemele logice din prima lecție și ca C++-ul de la lecțiile 4–6. Cine știe să **citească** un algoritm și să-l transforme în cod are deja jumătate din proba de BAC; restul sunt tehnici (vectori, funcții, grafuri) pe care le adaugi în anii următori.

2. Notațiile din pseudocodul de la Bac

Pseudocod	C++
`citește a, b`	`cin >> a >> b;`
`scrie x, " "`	`cout << x << " ";`
`x ← e`	`x = e;`
`[x]` (partea întreagă)	`x / y` între întregi sau `(int)x`
`x % y`	`x % y`
`=`, `≠`, `≤`, `≥`	`==`, `!=`, `<=`, `>=`
`și`, `sau`, `not`	`&&`, `||`, `!`
`┌dacă C atunci … altfel … ┘■`	`if (C) { … } else { … }`
`┌cât timp C execută … ┘■`	`while (C) { … }`
`┌repetă … ┘până când C`	`do { … } while (!C);`
`┌pentru i ← a, b, p execută … ┘■`	`for (i = a; i <= b; i += p) { … }` (pentru p > 0)

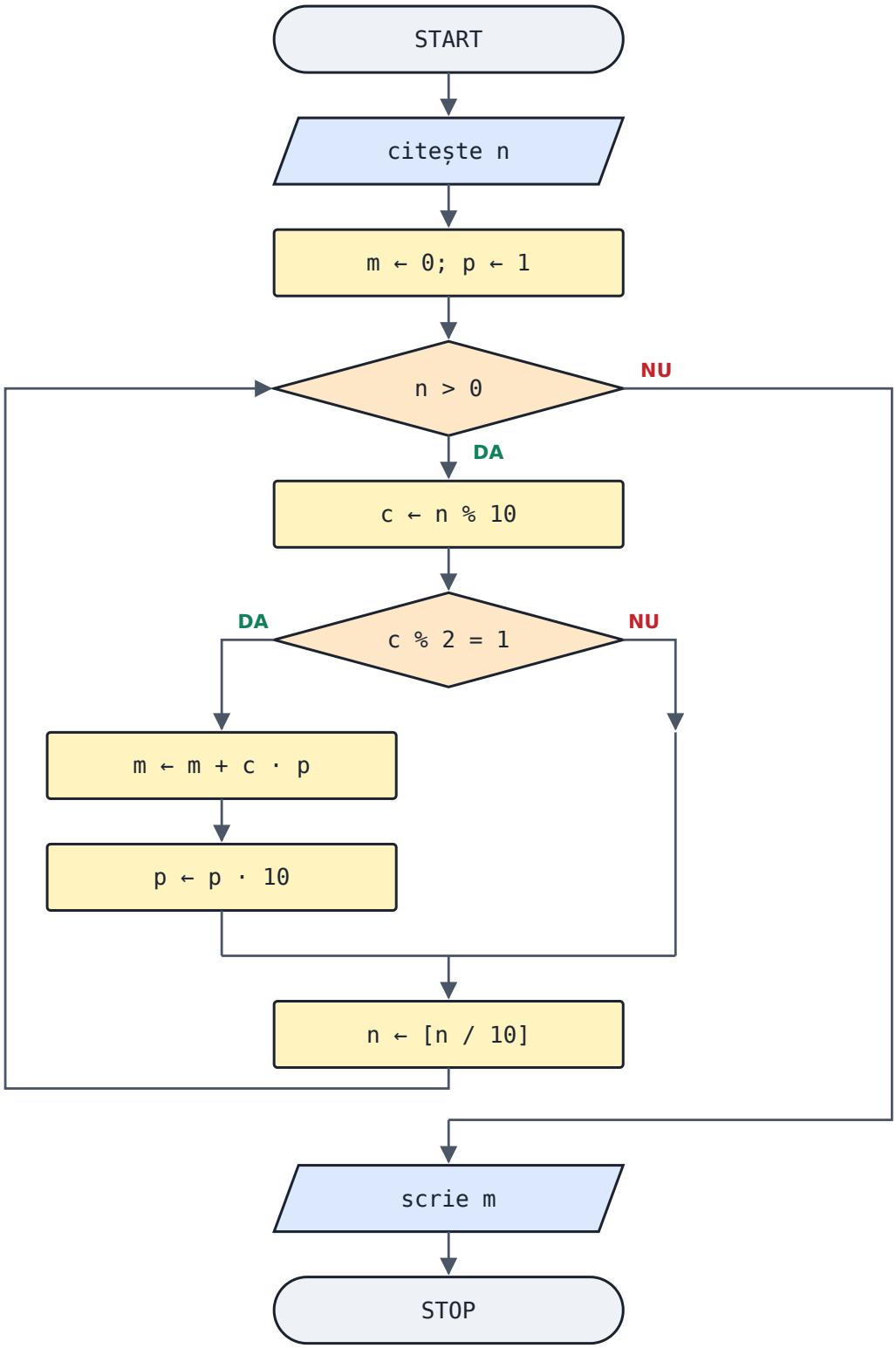
Liniile verticale `|` arată **ce instrucțiuni sunt în interiorul** unei structuri. Citește-le cu atenție: o singură instrucțiune pusă greșit înăuntru/în afara buclei schimbă rezultatul.

ATENȚIE — GREȘELI FRECVENTE

În pseudocod `=` este **comparație**, iar atribuirea este `←`. Când traduci în C++ nu scrie niciodată `=` într-un `if`; acolo ai nevoie de `==`.

3. Exemplu complet de tip II.1

```
citește n                          (număr natural)
m ← 0
p ← 1
┌cât timp n > 0 execută
│   c ← n % 10
│   ┌dacă c % 2 = 1 atunci
│   │   m ← m + c * p
│   │   p ← p * 10
│   └■
│   n ← [n / 10]
└■
scrie m
```



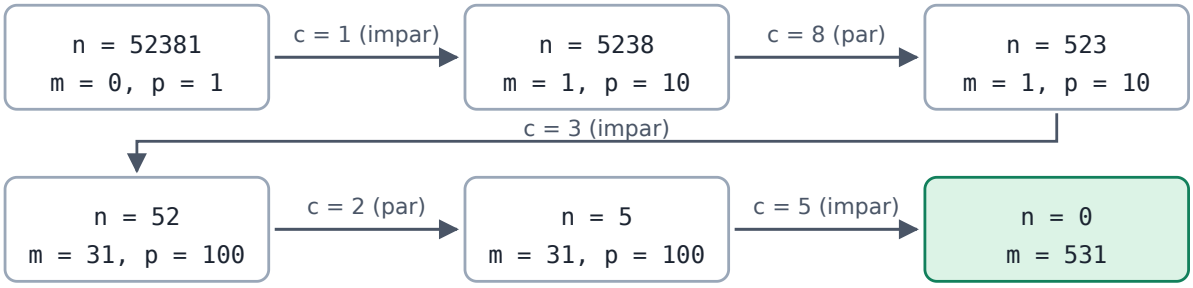
Păstrează cifrele impare, în aceeași ordine

Schema logică a algoritmului

a) Ce se afișează pentru n = 52381?

Metoda: **tabel de urmărire**. O coloană pentru fiecare variabilă și un rând pentru fiecare iterație.

n	c	c impar?	m	p
52381	—	—	0	1
52381 → 5238	1	da	0 + 1·1 = 1	10
5238 → 523	8	nu	1	10
523 → 52	3	da	1 + 3·10 = 31	100
52 → 5	2	nu	31	100
5 → 0	5	da	31 + 500 = 531	1000



Starea variabilelor

Se afișează **531**. Observăm: algoritmul păstrează cifrele **impare**, în aceeași ordine.

DE CE AȘA?

De ce `p ← p * 10`? `p` este „poziția" în numărul nou: 1 (unități), 10 (zeci), 100 (sute)... Fiecare cifră impară găsită se pune la poziția `p`, iar apoi `p` crește de 10 ori ca următoarea cifră să intre mai la stânga. Aceasta este aceeași idee ca la „oglindit" din lecția despre cifre.

SFAT

După ce faci tabelul, formulează în cuvinte **ce face** algoritmul („păstrează cifrele impare"). Te ajută enorm la punctele b) și d).

b) „Scrieți două valori distincte, cu exact 4 cifre, pentru care se afișează 35."

Lucrăm **invers**: cifrele impare ale lui n, în ordine de la dreapta la stânga, trebuie să fie `5` apoi `3` (pentru că 35 are 5 la unități), iar celelalte două cifre pare. Exemple: **3254** și **2305** (sau 3520, 8345...).

n	cifre de la dreapta	cifre impare păstrate	m
3254	4, 5, 2, 3	5, 3	5 + 3·10 = **35**
2305	5, 0, 3, 2	5, 3	**35**

ATENȚIE — GREȘELI FRECVENTE

Verifică fiecare răspuns trecându-l din nou prin algoritm. La b) se punctează doar valorile **complet corecte** și care respectă toate condițiile (exact 4 cifre, distincte etc.).

c) Programul C++ corespunzător

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

int main()
{
    int n, m = 0, p = 1, c;
    cin >> n;
    while (n > 0)
    {
        c = n % 10;
        if (c % 2 == 1)
        {
            m = m + c * p;
            p = p * 10;
        }
        n = n / 10;
    }
    cout << m;
    return 0;
}
```

Se punctează separat: declararea variabilelor, citirea, afișarea, fiecare structură (instrucțiunile repetitive și alternative), atribuirile și **corectitudinea globală** a programului.

DE CE AȘA?

De ce `[n / 10]` devine pur și simplu `n / 10` ? Cum `n` e întreg, `/` taie deja partea fracționară: este exact „partea întreagă". Dacă ar fi fost `double`, aveai nevoie de `(int)`.

d) Algoritm echivalent cu altă structură repetitivă

Înlocuim `cât timp` cu `repetă … până când`. Deoarece `repetă` execută corpul cel puțin o dată, punem un `dacă` în față:

```
citește n
m ← 0
p ← 1
┌dacă n > 0 atunci
│   ┌repetă
│   │   c ← n % 10
│   │   ┌dacă c % 2 = 1 atunci
│   │   │   m ← m + c * p
│   │   │   p ← p * 10
│   │   └■
│   │   n ← [n / 10]
│   └până când n ≤ 0
└■
scrie m
```

> **DE ȘTIUT**
>
> Aici, pentru n = 0, corpul nu are niciun efect (c = 0 e par, n rămâne 0), deci `dacă` -ul se poate omite. Dar dacă nu ești sigur, păstrează-l: e mereu corect.

4. Strategii pentru „găsiți valori pentru care...”

1. Înțelege ce calculează algoritmul (din tabelul de la a).
2. Lucrează **invers**: ce trebuie să fie adevărat ca să iasă rezultatul cerut?
3. Construiește cea mai simplă valoare posibilă, apoi variază-o pentru a doua valoare.
4. **Verifică** fiecare valoare.

5. Lucrează singur: încă două exerciții II.1

Răspunsurile (inclusiv programele C++) sunt la finalul lecției. Începe cu tabelul de urmărire.

Exercițiul A

```
citește a, b                (numere naturale nenule)
k ← 0
┌cât timp a ≥ b execută
│   a ← a - b
│   k ← k + 1
└■
scrie k, " ", a
```

a) Ce se afișează pentru `a = 23`, `b = 5`?
b) Ce calculează algoritmul? (indiciu: legătura cu `/` și `%`)
c) Scrie programul C++.
d) Scrie un algoritm echivalent care folosește `repetă … până când`.
e) Câte valori ale lui `a` din intervalul [1, 50] fac ca, pentru `b = 7`, a doua valoare afișată să fie 0?

Exercițiul B

```
citește n                   (număr natural)
k ← 0
┌cât timp n ≥ 10 execută
│   ┌dacă n % 10 < [n / 10] % 10 atunci
│   │   k ← k + 1
│   └■
│   n ← [n / 10]
└■
scrie k
```

a) Ce se afișează pentru `n = 95312`?
b) Dați un exemplu de număr cu 4 cifre pentru care se afișează 0.
c) Dați două numere distincte de 4 cifre pentru care se afișează 3.
d) Scrieți programul C++ și un algoritm echivalent cu `pentru`.
e) Ce calculează algoritmul? Formulați în cuvinte.

Exercițiul C

```
citește n                    (număr natural nenul)
s ← 0
┌pentru i ← 1, n execută
│   ┌dacă i % 2 = 0 atunci
│   │   s ← s + i
│   │altfel
│   │   s ← s - i
│   └■
└■
scrie s
```

a) Ce se afișează pentru `n = 5`? Dar pentru `n = 8`?
b) Pentru ce valoare a lui `n` se afișează 5?
c) Scrie programul C++.
d) Înlocuiți `pentru` cu `cât timp`.

6. Exersează pe hârtie (grile)

1. Ce valoare are `[x]` pentru `x = 7.9`? Dar `[7 / 2]`?

2. Care este traducerea corectă în C++ a lui `repetă … până când x ≤ 0`?

a) `do { … } while (x <= 0);`
b) `do { … } while (x > 0);`
c) `while (x > 0) { … }`
d) `while (x <= 0) { … }`

3. Un algoritm are pseudocodul `┌cât timp a ≠ b execută … └■`. Cum se scrie în C++ condiția?

4. După execuția secvenței ce valoare are `z`?

```
x ← 17
z ← 0
┌cât timp x > 0 execută
│   z ← z + x % 2
│   x ← [x / 2]
└■
```

7. Mai departe (pentru cine vrea mai mult)

Algoritmul de la 4. face ceva celebru. `z` numără cifrele de 1 din scrierea în baza 2 a lui `x`. Pentru 17 = 10001_2 sunt două cifre de 1, deci `z = 2`. Vei revedea ideea la lecția despre baze de numerație și la operatorii pe biți.

Cum se punctează. Baremul oficial dă puncte parțiale: citire, afișare, fiecare structură, corectitudinea globală. Chiar dacă nu găsești răspunsul la b), scrie un program c) corect: ia punctele pe care le poate lua. Strategia „nu las nimic necompletat” aduce puncte în plus.

Din pseudocod în cod real. În practică treci zilnic de la o specificație („păstrează cifrele impare”) la cod. Pseudocodul de la BAC e antrenamentul: mai întâi ce vrei să faci, apoi cum se scrie.

Rezumat

- II.1 = a) ce afișează, b) valori pentru un rezultat, c) program C++, d) structură echivalentă.
- Tabelul de valori + formularea în cuvinte a ce face algoritmul.
- `cât timp` → `dacă` + `repetă … până când` cu condiția **negată**.
- În pseudocod `←` atribuie și `=` compară; în C++ `=` atribuie și `==` compară.

Probleme de lucru (pbinfo)

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

- [ ] #3985 **suma-c2** — ușoară
  https://www.pbinfo.ro/probleme/3985/suma-c2
- [ ] #3992 **count-c2** — ușoară
  https://www.pbinfo.ro/probleme/3992/count-c2
- [ ] #4279 **sumcnt2** — ușoară
  https://www.pbinfo.ro/probleme/4279/sumcnt2

Temă (pbinfo)

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

- [ ] #1362 **p10** — ușoară
  https://www.pbinfo.ro/probleme/1362/p10
- [ ] #1565 **nzero** — ușoară
  https://www.pbinfo.ro/probleme/1565/nzero
- [ ] #4547 **sumcnt3** — ușoară
  https://www.pbinfo.ro/probleme/4547/sumcnt3

Grile din subiectele de BAC (pbinfo)

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

- [ ] #798 **Exercițiul #798**
  pseudocod cu a%b în buclă (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=798
- [ ] #800 **Exercițiul #800**
  pseudocod cu a%b în buclă (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=800
- [ ] #806 **Exercițiul #806**
  pseudocod cu a%b în buclă (Bac 2017)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=806

- [ ] #810 **Exercițiul #810**
  pseudocod cu a%b în buclă (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=810
- [ ] #811 **Exercițiul #811**
  pseudocod cu a%b în buclă (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=811
- [ ] #812 **Exercițiul #812**
  pseudocod cu a%b în buclă (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=812
- [ ] #1208 **Exercițiul #1208**
  pseudocod: ce se afișează (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1208
- [ ] #1209 **Exercițiul #1209**
  pseudocod: ce se afișează, variantă (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1209
- [ ] #1291 **Exercițiul #1291**
  citește n, x, y: ce calculează (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1291
- [ ] #1412 **Exercițiul #1412**
  pseudocod cu a%b (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1412
- [ ] #1413 **Exercițiul #1413**
  pseudocod cu a%b, variantă (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1413

Citește și exersează în plus

- Subiecte de la examene (Bac) https://www.pbinfo.ro/articole/16/subiecte-de-la-examene
- Subiecte de Bacalaureat — informatică (edu.ro) — portalul oficial cu subiectele și baremele din anii anteriori (verifică la secțiunea Bacalaureat) https://subiecte.edu.ro/
- Python Tutor pentru C++ — verifică tabelul tău: rulează programul pas cu pas
  https://pythontutor.com/cpp.html
- pbinfo — probleme de pseudocod / algoritmi — toate grilele de BAC, cu verificare automată
  https://www.pbinfo.ro/?pagina=itemi-evaluare-lista

Răspunsuri

Exercițiul A

a) `4 3`. a: 23 → 18 → 13 → 8 → 3; k = 4.

a	a ≥ 5 ?	k
23	da	0 → 1
18	da	2
13	da	3
8	da	4
3	nu → stop	4

b) Scade pe `b` din `a` cât timp se poate: `k = a / b` (câtul) și rămâne `a % b` (restul).

c)

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

int main()
{
    int a, b, k = 0;
    cin >> a >> b;
    while (a >= b)
    {
        a = a - b;
        k = k + 1;
    }
    cout << k << " " << a;
    return 0;
}
```

d)

```
citește a, b
k ← 0
┌dacă a ≥ b atunci
│  ┌repetă
│  │   a ← a - b
│  │   k ← k + 1
│  └până când a < b
└■
scrie k, " ", a
```

e) 7 valori: `a` trebuie să fie multiplu de 7 din [1, 50], adică 7, 14, 21, 28, 35, 42, 49.

Exercițiul B

a) `3`. Perechile (de la dreapta spre stânga): 2 față de 1: nu; 1 față de 3: da; 3 față de 5: da; 5 față de 9: da. k = 3.

b) De exemplu `1234` (nicio cifră nu e mai mică decât vecina din stânga: cifrele cresc de la stânga la dreapta).

c) De exemplu `9876` și `4321` (fiecare cifră e mai mică decât cea din stânga).

d)

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

int main()
{
    int n, k = 0;
    cin >> n;
    while (n >= 10)
    {
        if (n % 10 < n / 10 % 10)
            k++;
        n = n / 10;
    }
    cout << k;
    return 0;
}
```

Cu `pentru` nu se poate scrie direct (numărul de pași depinde de numărul de cifre); se poate însă cu `repetă … până când n < 10`, precedat de `dacă n ≥ 10`.

e) Numără câte cifre sunt **mai mici decât cifra din stânga lor**, adică de câte ori scade numărul pe parcurs de la stânga la dreapta.

Exercițiul C

a) Pentru `n = 5`: −1 + 2 − 3 + 4 − 5 = **−3**. Pentru `n = 8`: (−1 + 2) + (−3 + 4) + (−5 + 6) + (−7 + 8) = **4**.

i	1	2	3	4	5
s	−1	1	−2	2	−3

b) Pentru n par, s = n / 2; pentru n impar, s = −(n + 1) / 2 (mereu negativ). Se afișează 5 doar pentru **n = 10**.

c)

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

int main()
{
    int n, s = 0;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        if (i % 2 == 0)
            s = s + i;
        else
            s = s - i;
    }
    cout << s;
    return 0;
}
```

d)

```
citește n
s ← 0
i ← 1
┌cât timp i ≤ n execută
│   ┌dacă i % 2 = 0 atunci
│   │   s ← s + i
│   │altfel
│   │   s ← s - i
│   └■
│   i ← i + 1
└■
scrie s
```

Exersează pe hârtie (grile)

1. `[7.9] = 7` și `[7 / 2] = 3` (partea întreagă a lui 3,5).

2. b). `repetă … până când x ≤ 0` continuă cât timp `x > 0`: condiția se neagă. Greșeala clasică este a).

3. `a != b`.

4. `z = 2`. x: 17 → 8 → 4 → 2 → 1 → 0, iar `x % 2` ia valorile 1, 0, 0, 0, 1: suma este 2.