INFORMATICĂ · CLASA A IX-A · ALGORITMI ELEMENTARI · LECȚIA 8

Algoritmi elementari: sume, produse, numărări, maxim și minim

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-06

CE VEI ȘTI DUPĂ LECȚIE

- Calculezi sume și produse după o formulă dată (S = 1·2 + 2·3 + … , n! etc.).
- Numeri elementele care au o proprietate.
- Determini maximul / minimul unui șir și de câte ori apare.
- Determini cele mai mari două valori dintr-un șir, într-o singură parcurgere.
- Alegi tipul de date potrivit ca să eviți depășirea.

1. Ideea: rezolvăm „din mers”

Majoritatea problemelor cu șiruri de numere au aceeași structură: **citești un număr, îl treci prin ceva și mergi mai departe**. Nu ai nevoie să ții toate numerele minte, ci doar un **rezumat** (o sumă, un contor, un maxim) pe care îl actualizezi la fiecare număr. În lecția aceasta sunt cele patru rezumate de bază: sumă, produs, contor, maxim/minim.

2. Sume și produse după o formulă

Rețeta:

1. Identifici **termenul general** (al i-lea termen) ca formulă de `i`.
2. Inițializezi suma cu `0` (produsul cu `1`).
3. Parcurgi `i` de la 1 la n și aduni (înmulțești) termenul.

Expresie	Termenul i	Cod în buclă
1 + 2 + … + n	`i`	`s += i;`
$1^2 + 2^2 + \dots + n^2$	`i * i`	`s += i * i;`
1·2 + 2·3 + … + n(n+1)	`i * (i + 1)`	`s += i * (i + 1);`
1 − 2 + 3 − 4 …	`semn * i`	`s += semn * i; semn = -semn;`
n! = 1·2·…·n	`i`	`p *= i;`

```
long long s = 0;
for (int i = 1; i <= n; i++)
    s += 1LL * i * (i + 1);     // 1LL forțează calculul în long long
```

Urmărire pentru `n = 4`, suma `1·2 + 2·3 + 3·4 + 4·5`:

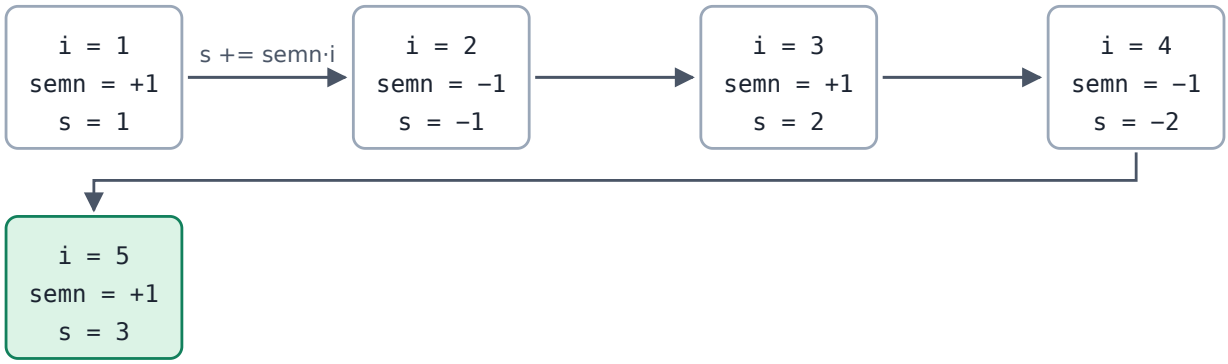
i	termenul `i·(i+1)`	s după
(început)		0
1	2	2
2	6	8
3	12	20
4	20	40

DE CE AȘA?

De ce suma pornește de la 0 și produsul de la 1? Sunt elementele **neutre**: 0 nu schimbă o sumă, 1 nu schimbă un produs. Un produs pornit de la 0 ar rămâne 0 oricâte numere ai înmulți.

Semne alternante

Pentru `1 − 2 + 3 − 4 + …` folosim o variabilă `semn` care își schimbă valoarea:



semn = −semn la sfârșitul fiecărei iterații

Suma alternantă

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

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

DE CE AȘA?

De ce `semn` și nu `if (i % 2 == 0)`? Ambele merg. Varianta cu `semn` se adaptează imediat la tipare mai greu de exprimat cu `%` (de exemplu +, +, −, −) și evită un `if` în buclă. La BAC vei întâlni ambele forme: recunoaște-le.

ATENȚIE — GREȘELI FRECVENTE

Depășirea: 13! depășește deja `int`, iar 21! depășește chiar și `long long`. Citește restricțiile din enunț și calculează pe hârtie cât de mare poate fi rezultatul. `1LL * a * b` face înmulțirea pe 64 de biți.

Termen care depinde de cel anterior

Pentru `S = 1 + 1·2 + 1·2·3 + … + 1·2·…·n` nu recalculezi fiecare factorial: îl **actualizezi**:

```
long long p = 1, s = 0;
for (int i = 1; i <= n; i++)
{
    p *= i;      // p = i!
    s += p;
}
```

i	p (= i!)	s
1	1	1
2	2	3
3	6	9
4	24	33

DE CE AȘA?

De ce actualizăm, nu recalculăm? Dacă pentru fiecare `i` ai calcula `i!` de la zero, ai face 1 + 2 + … + n înmulțiri (aproximativ $n^2/2$). Actualizând `p *= i` faci doar n. Ideea „noul termen se obține din cel vechi" apare în aproape orice problemă cu șiruri (Fibonacci, puteri, factorial).

SFAT

Formulele lui Gauss te scapă de bucle: $1 + 2 + \ldots + n = n(n+1)/2$ și $1^2 + \ldots + n^2 = n(n+1)(2n+1)/6$. La BAC, la problema de eficiență, o formulă e întotdeauna mai rapidă decât o buclă.

3. Numărări

```
int cnt = 0;
for (int i = 1; i <= n; i++)
{
    cin >> x;
    if (x % 2 == 0) cnt++;          // condiția = proprietatea cerută
}
```

Pentru **perechi de elemente consecutive** păstrezi valoarea anterioară:

```
cin >> ant;
for (int i = 2; i <= n; i++)
{
    cin >> x;
    if ((ant + x) % 2 == 0) cnt++;   // perechi consecutive cu suma pară
    ant = x;
}
```

Urmărire pentru șirul `3 5 8 2`:

i	ant	x	`(ant + x) % 2 == 0` ?	cnt
(început)	3			0
2	3	5	da (8)	1
3	5	8	nu (13)	1
4	8	2	da (10)	2

DE CE AȘA?

De ce bucla începe de la `i = 2`? Primul număr a fost deja citit în `ant`. Pentru n numere există n − 1 perechi consecutive, deci n − 1 iterații. Greșeala frecventă este un „pas în plus" sau „unul în minus": verifică mereu pe un exemplu cu n = 2.

4. Maximul și minimul

Ideea: presupunem că primul element este maximul, apoi îl comparăm cu fiecare element nou.

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

int main()
{
    int n, x;
    cin >> n >> x;
    int mx = x, mn = x;              // primul număr citit
    for (int i = 2; i <= n; i++)
    {
        cin >> x;
        if (x > mx) mx = x;
        if (x < mn) mn = x;
    }
    cout << mx << " " << mn;
    return 0;
}
```

Intrare / ieșire:

Intrare	Ieșire	Explicație
`5` / `4 9 6 9 2`	`9 2`	maximul 9, minimul 2

Intrare	Ieșire	Explicație
`3` / `-5 -2 -9`	`-2 -9`	cu numere negative inițializarea cu primul element merge

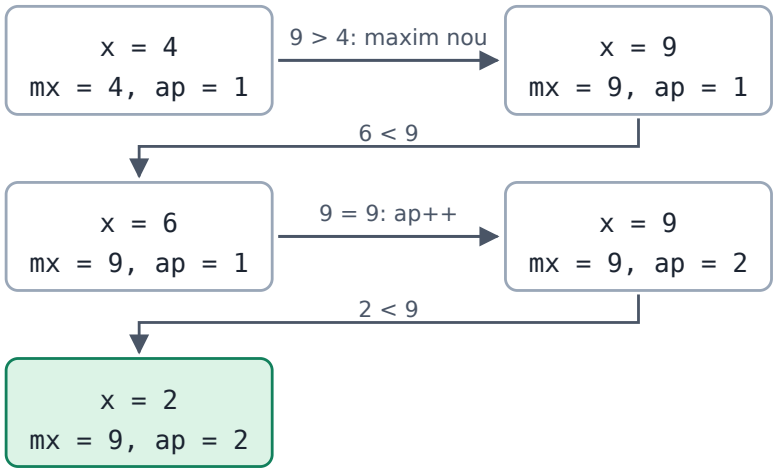
CAPCANĂ

Inițializarea `mx = 0` este greșită dacă numerele pot fi negative (maximul lui −5, −2, −9 ar ieși 0). Inițializează **cu primul element** sau cu o valoare mai mică decât orice valoare posibilă (de exemplu `INT_MIN` din `<climits>`, sau `-1` dacă numerele sunt naturale).

DE CE AȘA?

De ce primul element și nu o valoare „foarte mică”? Maximul trebuie să fie un număr din șir. Dacă pornești de la primul element, ai garanția că răspunsul e unul dintre numerele citite, indiferent de domeniul valorilor. O valoare artificială (de tip `-1000000`) nu e sigură: apare un test cu −2 000 000 și programul pică.

Maximul și de câte ori apare



Maxim și apariții

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

int main()
{
    int n, x, mx = -1, ap = 0;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> x;
        if (x > mx) { mx = x; ap = 1; }   // maxim nou → îl numărăm de la 1
        else if (x == mx) ap++;           // încă o apariție
    }
    cout << mx << " " << ap;
    return 0;
}
```

x	`x > mx` ?	`x == mx` ?	mx	ap
(început)			−1	0
4	da		4	1
9	da		9	1
6	nu	nu	9	1
9	nu	da	9	2
2	nu	nu	9	2

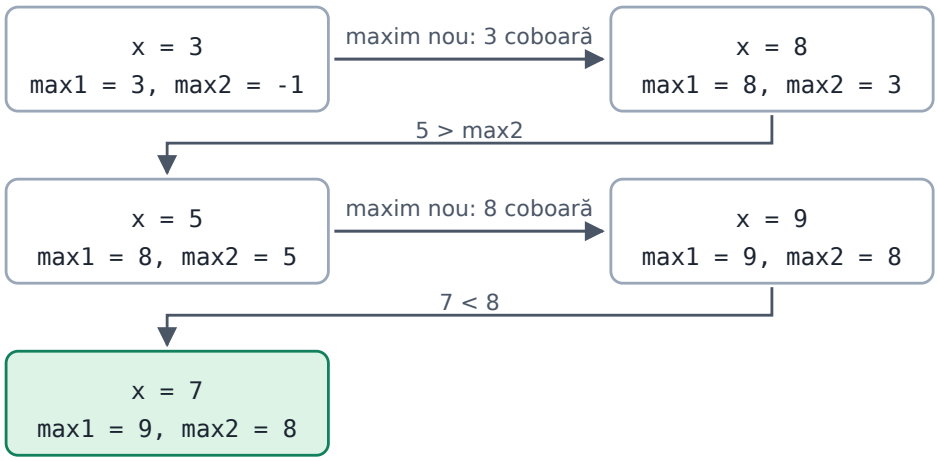
DE CE AȘA?

De ce `ap = 1` când apare un maxim nou? Contorul vechi se referea la **vechiul** maxim, care nu mai contează. Maximul nou a apărut o singură dată (până acum). Tiparul „când schimb răspunsul, resetez și contorul asociat" se folosește la orice „valoarea cea mai... și de câte ori".

Poziția maximului: prima sau ultima apariție?

- `if (x > mx)` → reține **prima** poziție a maximului;
- `if (x >= mx)` → reține **ultima** poziție.

Cele mai mari două valori (într-o parcurgere)



Două maxime

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

int main()
{
    int n, x, max1 = -1, max2 = -1;      // max1 ≥ max2
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> x;
        if (x > max1) { max2 = max1; max1 = x; }   // vechiul maxim coboară pe locul 2
        else if (x > max2) max2 = x;
    }
    cout << max1 << " " << max2;
    return 0;
}
```

DE CE AȘA?

De ce `max2 = max1` *înainte* de `max1 = x` ? Dacă ai face invers, ai pierde vechiul maxim înainte să-l poți salva. Ordinea instrucțiunilor într-un „schimb de roluri” contează întotdeauna: salvezi, apoi suprascrii. (Pentru `4 9 9 3` programul de mai sus afișează `9 9` : valorile egale se numără separat. Dacă enunțul cere două valori **distincte**, condiția a doua devine `x > max2 && x < max1` .)

LA BAC

„Cele mai mari două valori” sau „primele două numere impare” sunt teme clasice pentru Subiectul III.3 (algoritm eficient, fără vectori). Varianta de mai sus face o singură parcurgere și folosește memorie constantă: exact ce se cere pentru punctajul maxim la eficiență.

5. Greșeli frecvente

Greșeala	Efect	Corect
`mx = 0` la numere negative	maximul ieșit 0	`mx = primul` sau `INT_MIN`
`s` neinițializat	rezultat aleator	`s = 0`
produs inițializat cu 0	produsul rămâne 0	`p = 1`
`int` pentru factorial mare	depășire, valori negative	`long long`
contor neresetat la maxim nou	`ap` prea mare	`ap = 1` la `x > mx`
`i` începe de la 1 la perechi consecutive	citești o valoare în plus	începe de la 2

6. La BAC

> **LA BAC**
>
> Aceste tipare apar la **Subiectul III** (programe fără vectori, cu un șir de numere dat) și în grile cu „ce valoare are variabila `s` / `mx` după secvență". Rezolvarea grilelor: tabel cu coloanele variabilelor (ca în lecție). Rezolvarea problemei: alege rezumatul (sumă/contor/maxim) și scrie bucla.

7. Exercițiu de gândire (rezolvare la final)

Se citesc `n` numere naturale. Afișează **suma dintre cel mai mare și cel mai mic număr par**. Dacă nu există numere pare, afișează `NU EXISTA`. Gândește: cu ce inițializezi? Cum detectezi că nu există niciun număr par?

8. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Ce valoare are `s` după `s = 0; for (i = 1; i <= 5; i++) s += i * i;` ?

2. Ce valoare are `p` după `p = 1; for (i = 2; i <= 5; i++) p *= i;` ?

3. Câte numere din intervalul [1, 20] sunt divizibile cu 3 sau cu 5?

```
cnt = 0;
for (i = 1; i <= 20; i++)
    if (i % 3 == 0 || i % 5 == 0) cnt++;
```

4. Ce afișează secvența pentru intrarea `3 7 7 2 7` ?

```cpp
int mx = -1, ap = 0, x;
for (int i = 1; i <= 5; i++)
{
    cin >> x;
    if (x > mx) { mx = x; ap = 1; }
    else if (x == mx) ap++;
}
cout << mx << ' ' << ap;
```

5. Ce afișează, pentru `n = 4` și numerele `4 9 9 3`, programul cu `max1` / `max2` din lecție?

6. Pentru numerele `-5 -2 -9`, ce afișează secvența următoare? Este corect?

```cpp
int mx = 0;
for (int i = 1; i <= 3; i++) { cin >> x; if (x > mx) mx = x; }
cout << mx;
```

7. Calculează fără buclă `1·2 + 2·3 + … + 10·11`, știind că suma `1·2 + … + n·(n+1)` este `n(n+1)(n+2)/3`.

8. Care este cel mai mare `n` pentru care `n!` încape într-un `int` de 32 de biți (maxim 2 147 483 647)? Dar într-un `long long` (maxim aproximativ $9{,}2 \cdot 10^{18}$)?

9. Scrie secvența care calculează media aritmetică a celor `n` numere citite, afișată cu 2 zecimale.

9. Mai departe (pentru cine vrea mai mult)

Suma Gauss, demonstrația. Scrie 1 + 2 + ... + n și, dedesubt, același șir invers. Fiecare coloană are suma n + 1, iar coloanele sunt n, deci dublul sumei este n(n + 1). De la aici vine n(n+1)/2.

Medii și variații. Cu doar două rezumate (suma și numărul de elemente) poți calcula media; cu încă unul (suma pătratelor) poți calcula și **dispersia**, o idee din statistică folosită în toate aplicațiile cu date (analiza de scoruri, senzori).

Algoritmul lui Kadane. „Cea mai mare sumă a unei secvențe consecutive" se rezolvă tot într-o parcurgere, cu două rezumate: suma curentă și cel mai bun maxim văzut. Este o problemă celebră de interviu; cu tiparul de azi ești la un pas de ea.

Moduri de a depăși. Dacă rezultatul nu încape nici în `long long`, se calculează **modulo** un număr (de exemplu $10^9 + 7$): `s = (s + termen) % MOD;`. Apare la concursuri, aproape în fiecare an.

Rezumat

- Sumă: start `0`; produs: start `1`; termenul general scris în funcție de `i`.
- Atenție la depășire: `long long`, `1LL * a * b`.
- Numărare: `cnt = 0` + `if (proprietate) cnt++`.
- Maxim/minim: inițializează cu primul element; la un maxim nou, resetează contorul de apariții.
- Două maxime: când apare un maxim nou, vechiul maxim devine al doilea.

Probleme de lucru (pbinfo)

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

- [ ] #46 **sumapare** — ușoară
  S = 2 + 4 + ... + 2n. Poți și cu formulă: n(n+1).
  https://www.pbinfo.ro/probleme/46/sumapare
- [ ] #49 **factorial** — ușoară
  Produs: inițializare cu 1.
  https://www.pbinfo.ro/probleme/49/factorial
- [ ] #354 **n-maxim** — ușoară
  Maximul a n numere întregi (pot fi și negative!).
  https://www.pbinfo.ro/probleme/354/n-maxim
- [ ] #346 **maxandap** — ușoară
  Maximul și numărul lui de apariții, într-o singură parcurgere.
  https://www.pbinfo.ro/probleme/346/maxandap

Temă (pbinfo)

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

- [ ] #47 **sumapatrate** — ușoară
  https://www.pbinfo.ro/probleme/47/sumapatrate
- [ ] #332 **expresie2** — ușoară
  https://www.pbinfo.ro/probleme/332/expresie2
- [ ] #335 **expresie3** — ușoară
  Semne alternante: folosește o variabilă semn = ±1.
  https://www.pbinfo.ro/probleme/335/expresie3
- [ ] #347 **summaxmin** — ușoară
  https://www.pbinfo.ro/probleme/347/summaxmin
- [ ] #4280 **maxdivk** — ușoară
  https://www.pbinfo.ro/probleme/4280/maxdivk
- [ ] #3980 **minplusmax** — ușoară
  https://www.pbinfo.ro/probleme/3980/minplusmax
- [ ] #119 **2maxim** — medie
  Cele mai mari două valori — fără să sortezi.
  https://www.pbinfo.ro/probleme/119/2maxim
- [ ] #50 **expresie** — medie
  S = 1 + 1·2 + 1·2·3 + ... : produsul se actualizează din mers.
  https://www.pbinfo.ro/probleme/50/expresie

Pentru cei care vor mai mult

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

- [ ] #274 **3minime** — medie
  https://www.pbinfo.ro/probleme/274/3minime
- [ ] #172 **difmin** — medie
  https://www.pbinfo.ro/probleme/172/difmin

Grile din subiectele de BAC (pbinfo)

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

- [ ] #1262 **Exercițiul #1262**
  maxime și minime într-o secvență (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1262
- [ ] #1621 **Exercițiul #1621**
  pseudocod cu min_v și max_v
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1621
- [ ] #1053 **Exercițiul #1053**
  secvență cu numere naturale (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1053
- [ ] #1062 **Exercițiul #1062**
  secvență cu numere naturale, variantă (Bac 2020)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1062
- [ ] #1734 **Exercițiul #1734**
  valoarea afișată pentru n = 64598 (Bac 2010)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1734
- [ ] #653 **Exercițiul #653**
  pseudocod: cifrele unui număr (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=653
- [ ] #654 **Exercițiul #654**
  pseudocod: cifrele unui număr, variantă (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=654

Citește și exersează în plus

- Maxime și minime https://www.pbinfo.ro/articole/79/maximul-minimul-pentru-un-numar-oarecare-de-valori
- Factorialul unui număr natural https://www.pbinfo.ro/articole/10944/factorial
- CSES Problem Set — Introductory Problems — probleme în engleză, cu teste ascunse; secțiunea Introductory se potrivește după această lecție https://cses.fi/problemset/

- Python Tutor pentru C++ — urmărește cum se schimbă mx și ap la fiecare element https://pythontutor.com/cpp.html
- cppreference — limitele tipurilor numerice — INT_MAX, LLONG_MAX: cât de mare poate fi un rezultat https://en.cppreference.com/w/cpp/types/numeric_limits

Răspunsuri

Exercițiu de gândire

Inițializezi `mxp = -1` și `mnp = -1` (valorile sunt naturale) și detectezi existența unui număr par prin `mxp == -1` la final.

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

int main()
{
    int n, x, mxp = -1, mnp = -1;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> x;
        if (x % 2 == 0)
        {
            if (mxp == -1 || x > mxp) mxp = x;
            if (mnp == -1 || x < mnp) mnp = x;
        }
    }
    if (mxp == -1) cout << "NU EXISTA";
    else cout << mxp + mnp;
    return 0;
}
```

Exersează pe hârtie

1. `55` (1 + 4 + 9 + 16 + 25).

2. `120` (2 · 3 · 4 · 5; la fel ca 5!, pentru că factorul 1 nu schimbă produsul).

3. `9`. Multiplii lui 3: 6 (3, 6, ..., 18); ai lui 5: 4 (5, 10, 15, 20); cei ai lui 15 sunt numărați de două ori (15): 6 + 4 − 1 = 9.

4. `7 3`. 3: mx = 3, ap = 1; 7: mx = 7, ap = 1; 7: ap = 2; 2: nimic; 7: ap = 3.

5. `9 9`. 4: max1 = 4; 9: max2 = 4, max1 = 9; 9: nu e `> max1`, dar e `> max2 (4)` → max2 = 9; 3: nimic. Două valori egale se numără separat.

6. Afișează `0`, ceea ce este **greșit**: maximul real este −2. Cauza: `mx` a fost inițializat cu 0, mai mare decât toate numerele. Corect: `mx = primul număr`.

7. 10 · 11 · 12 / 3 = **440**.

8. Într-un `int`: `n = 12` (12! = 479 001 600; 13! = 6 227 020 800 depășește). Într-un `long long`: `n = 20` (20! ≈ 2,43 · 10^{18}; 21! depășește).

9.

```cpp
long long s = 0;
for (int i = 1; i <= n; i++) { cin >> x; s += x; }
cout << fixed << setprecision(2) << (double)s / n;   // #include <iomanip>
```

Împărțirea trebuie făcută cu cel puțin un operand real: `(double)s / n`.