INFORMATICĂ · CLASA A IX-A · ALGORITMI ELEMENTARI · LECȚIA 9

Cifrele unui număr: sumă, oglindit, palindrom, construire de numere

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

CE VEI ȘTI DUPĂ LECȚIE

- Înțelegi de ce n % 10 dă ultima cifră și n / 10 o „taie", pornind de la scrierea zecimală.
- Parcurgi cifrele unui număr și calculezi suma, produsul, numărul de cifre, cifra maximă sau minimă.
- Construiești oglinditul și verifici dacă un număr este palindrom.
- Construiești un număr nou din anumite cifre, păstrând ordinea lor (metoda cu p = 1, 10, 100...).
- Tratezi corect cazurile-capcană: n = 0, zerourile de la final, depășirea tipului int.

1. De unde pornim: cum „arată" un număr în calculator

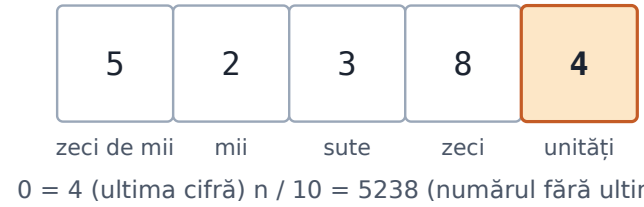
Când scrii `n = 52384`, calculatorul nu păstrează cinci cifre separate, ci **o singură valoare**. Nu poți cere direct „a treia cifră", ca într-un șir de litere. Avem însă două operații care, împreună, ne dau acces la fiecare cifră, una câte una: `%` (restul) și `/` (câtul împărțirii întregi).

Ideea vine din felul în care scriem numerele în baza 10:

52384 = 5 · 10000 + 2 · 1000 + 3 · 100 + 8 · 10 + 4

Toți termenii, în afară de ultimul, se împart exact la 10. Deci:

- **restul** împărțirii la 10 este exact ultima cifră: `52384 % 10 = 4`;
- **câtul** împărțirii la 10 este numărul fără ultima cifră: `52384 / 10 = 5238` (la `int`, partea zecimală se pierde).



Numărul 52384: ultima cifră se obține cu n % 10, restul numărului cu n / 10

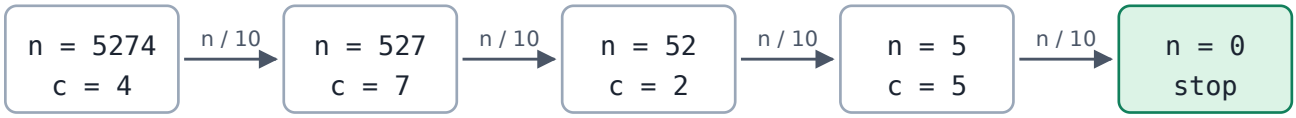
DE ȘTIUT

La fel, `n % 100` dă ultimele **două** cifre (`84`), iar `n / 100` taie două cifre (`523`). În general, `n % 10^k` păstrează ultimele k cifre, iar `n / 10^k` le taie.

2. Ideea de bază, fără cod

Gândește-te la număr ca la un teanc de cartonașe cu cifre. Nu putem lua decât cartonașul din dreapta. Ca să ajungem la toate:

1. ne uităm la cifra din dreapta (`n % 10`) și facem ce avem de făcut cu ea (o adunăm, o numărăm, o comparăm...);
2. o aruncăm (`n = n / 10`);
3. repetăm **cât timp** mai avem cifre, adică cât timp `n > 0` .



La fiecare pas: c = n % 10 (cifra din dreapta), apoi n = n / 10 (o tăiem)

Cum dispare câte o cifră din 5274: la fiecare pas c primește cifra din dreapta

Cifrele ies **de la dreapta la stânga**: 4, 7, 2, 5. Ține minte asta: e motivul pentru care oglinditul se construiește „de la sine", iar păstrarea ordinii inițiale cere un truc (secțiunea 6).

3. De la idee la cod: suma cifrelor

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

int main()
{
    int n, s = 0;
    cin >> n;
    while (n > 0)          // mai avem cifre?
    {
        int c = n % 10;    // cifra din dreapta
        s = s + c;         // o folosim
        n = n / 10;        // o tăiem
    }
    cout << s;
    return 0;
}
```

Urmărim pas cu pas, pentru n = 5274

Pasul	`c = n % 10`	`s` după adunare	`n` după tăiere
înainte de buclă	–	0	5274
1	4	4	527
2	7	11	52
3	2	13	5
4	5	18	0

La pasul 4, `n` devine 0, condiția `n > 0` e falsă și bucla se oprește. Se afișează 18. Bucla s-a executat de **4 ori, câte o dată pentru fiecare cifră**.

DE CE AȘA?

De ce `while` și nu `for`? Nu știm dinainte câte cifre are numărul. `while` se oprește singur când nu mai sunt cifre.

De ce `s = 0` la început? Suma unui grup gol de numere este 0; dacă uiți inițializarea, `s` pornește de la o valoare oarecare din memorie.

De ce `n > 0` și nu `n != 0`? Pentru numere naturale sunt echivalente; `n > 0` e varianta folosită în barem și citește mai clar „mai sunt cifre".

Exemple de date

Intrare	Ieșire	De ce contează exemplul
`5274`	`18`	cazul obișnuit
`9`	`9`	o singură cifră: bucla se execută o dată
`1000000`	`1`	multe zerouri: nu modifică suma, dar sunt parcurse
`0`	`0`	bucla nu se execută deloc; aici rezultatul e totuși corect

4. Același șablon, alte prelucrări

Schimbi doar **inițializarea** și **ce faci cu cifra** `c`. Restul buclei rămâne identic.

Ce calculăm	Înainte de buclă	În buclă, cu cifra `c`
suma cifrelor	`s = 0`	`s = s + c;`
produsul cifrelor	`p = 1`	`p = p * c;`
numărul de cifre	`nr = 0`	`nr++;`
cifra maximă	`mx = 0`	`if (c > mx) mx = c;`
cifra minimă	`mn = 9`	`if (c < mn) mn = c;`
câte cifre pare	`k = 0`	`if (c % 2 == 0) k++;`
de câte ori apare cifra `x`	`k = 0`	`if (c == x) k++;`
există o cifră egală cu `x`?	`gasit = false`	`if (c == x) gasit = true;`

DE CE AȘA?

De ce produsul pornește de la 1, iar suma de la 0? 0 este elementul neutru al adunării, 1 al înmulțirii. Dacă produsul ar porni de la 0, ar rămâne 0 orice ar urma.

De ce cifra minimă pornește de la 9? Orice cifră este ≤ 9, deci prima cifră întâlnită va înlocui sigur valoarea de start. Merge și `mn = 10`, sau `mn = n % 10` (prima cifră citită). Nu merge `mn = 0`: nicio cifră nu e mai mică decât 0, deci rezultatul ar fi mereu 0.

CAPCANĂ

Cazul n = 0. Pentru `n = 0`, bucla `while (n > 0)` nu se execută deloc: numărul de cifre iese 0 (corect ar fi 1), cifra minimă iese 9 (corect ar fi 0). Fie tratezi separat `n = 0`, fie folosești `do-while`, care execută corpul cel puțin o dată:

```
int nr = 0;
do
{
    nr++;
    n = n / 10;
} while (n > 0);
```

Prima cifră

Ca să rămâi cu prima cifră, tai cifre **cât timp numărul are cel puțin două cifre**, adică cât timp e mai mare decât 9:

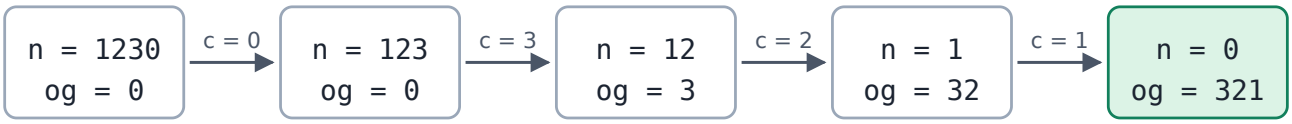
```
while (n > 9)
    n = n / 10;
// acum n este prima cifră
```

Pentru 5274: 5274 → 527 → 52 → 5, iar 5 nu mai e > 9, deci ne oprim. Merge și pentru n = 0 (rămâne 0).

5. Oglinditul și palindromul

Oglinditul lui 1234 este 4321. Cum cifrele ies deja de la dreapta la stânga (4, 3, 2, 1), le „lipim” pe rând la **dreapta** unui număr nou:

```
int og = 0, copie = n;
while (n > 0)
{
    og = og * 10 + n % 10;   // mutăm cifrele lui og spre stânga și punem cifra nouă la unități
    n = n / 10;
}
bool palindrom = (og == copie);
```



og = og * 10 + c: cifrele vechi se mută spre stânga, cifra nouă intră la unități

Construirea oglinditului lui 1230: la fiecare pas, og se înmulțește cu 10 și primește cifra nouă

Pasul	`c`	`og = og * 10 + c`	`n`
start	–	0	1230
1	0	0 · 10 + 0 = **0**	123
2	3	0 · 10 + 3 = **3**	12
3	2	3 · 10 + 2 = **32**	1
4	1	32 · 10 + 1 = **321**	0

DE ȘTIUT

Oglinditul lui 1230 este 321, nu 0321: zerourile din față nu se scriu. De aici rezultă că un număr care se termină în 0 (și nu e chiar 0) **nu poate fi palindrom**.

DE CE AȘA?

De ce păstrăm `copie`? Bucla distruge `n` (la final e 0). Ca să comparăm oglinditul cu numărul inițial, trebuie să-l fi salvat **înainte** de buclă. Aceeași greșeală apare în multe lucrări de BAC: comparația `og == n` după buclă e mereu falsă (pentru n ≠ 0).

ATENȚIE — GREȘELI FRECVENTE

Depășirea tipului `int`. `int` merge până la aproximativ 2,1 miliarde. Oglinditul lui 1 999 999 999 ar fi 9 999 999 991, care nu mai încape. Pentru numere cu 10 cifre folosește `long long` pentru `og`.

6. Construirea unui număr din anumite cifre

Problemele de BAC cer des: „construiește numărul format din cifrele pare ale lui n” sau „elimină cifrele egale cu x”. Există două variante, după ordinea cerută.

a) Ordine inversă: varianta simplă

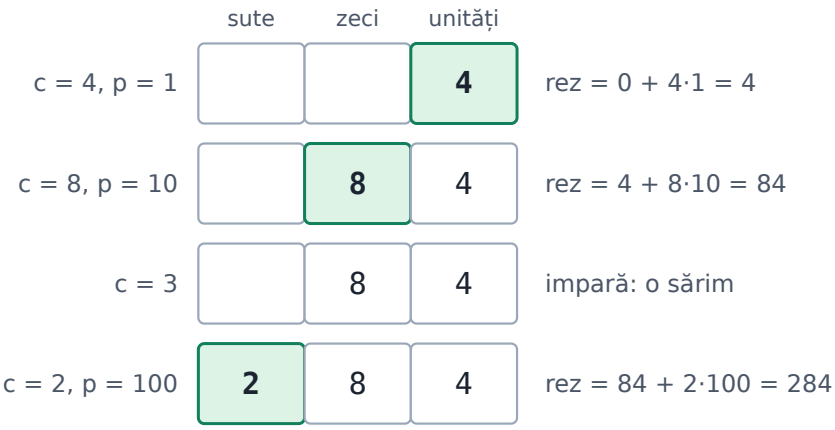
Ca la oglindit: fiecare cifră păstrată se lipește la dreapta. Din 52384, cifrele pare ies în ordinea 4, 8, 2, deci obținem `482`.

```
int rez = 0;
while (n > 0)
{
    int c = n % 10;
    if (c % 2 == 0)
        rez = rez * 10 + c;
    n = n / 10;
}
```

b) Aceeași ordine: metoda puterii lui 10

Vrem `284`, adică cifrele pare **în ordinea din număr**. Cifrele ies tot de la dreapta (4, 8, 2), dar de data asta fiecare cifră nouă trebuie pusă **în fața** celor de dinainte. Folosim `p`, care spune pe ce poziție intră cifra: unități (1), zeci (10), sute (100)...

```
int rez = 0, p = 1;
while (n > 0)
{
    int c = n % 10;
    if (c % 2 == 0)
    {
        rez = rez + c * p;   // cifra c intră pe poziția p
        p = p * 10;          // următoarea cifră păstrată va intra cu o poziție mai la
stânga
    }
    n = n / 10;
}
```



Fiecare cifră păstrată intră în fața celor de dinainte, pe poziția dată de p

Metoda puterii lui 10 pentru 52384: cifrele pare 4, 8, 2 intră pe unități, zeci, sute

Pasul	c	e pară?	rez	p după pas	n
1	4	da	0 + 4 · 1 = 4	10	5238
2	8	da	4 + 8 · 10 = 84	100	523
3	3	nu	84	100	52
4	2	da	84 + 2 · 100 = 284	1000	5
5	5	nu	284	1000	0

DE CE AȘA?

De ce `p` crește doar când păstrăm cifra? `p` nu urmărește poziția cifrei în numărul vechi, ci poziția **în numărul nou**. Dacă am înmulți `p` cu 10 la fiecare pas, cifrele sărite ar lăsa „găuri" de zerouri (am obține 20084 în loc de 284).

LA BAC

Metoda cu `p` apare în aproape fiecare sesiune de BAC: „eliminați cifrele...", „dublați cifrele pare...", „înlocuiți fiecare cifră impară cu...". Exersează-o până o scrii fără să te gândești.

c) Transformări: dublarea cifrelor pare

Fiecare cifră pară apare de două ori: 2345 → 223445. Fiecare cifră se pune o dată, iar dacă e pară se mai pune o dată:

```
int rez = 0, p = 1;
while (n > 0)
{
    int c = n % 10;
    rez = rez + c * p;  p = p * 10;
    if (c % 2 == 0)
    {
        rez = rez + c * p;  p = p * 10;
    }
    n = n / 10;
}
```

7. Cifra de control

Se adună cifrele, apoi cifrele sumei și tot așa, până rămâne o singură cifră: 9875 → 29 → 11 → 2.

Varianta directă, cu două bucle (una mare care se repetă cât timp avem cel puțin două cifre, una mică pentru suma cifrelor):

```
while (n > 9)
{
    int s = 0;
    while (n > 0) { s = s + n % 10; n = n / 10; }
    n = s;
}
// n este cifra de control
```

Există și o formulă de o linie, explicată în secțiunea „Mai departe": `control = (n == 0) ? 0 : 1 + (n - 1) % 9;`.

8. Greșeli frecvente

ATENȚIE — GREȘELI FRECVENTE

> - **Compari cu `n` după buclă.** La final `n` e 0. Salvează o copie înainte. - **Folosești `n % 10` de două ori în același pas, după ce ai tăiat deja cifra.** Pune cifra într-o variabilă `c` la începutul pasului și lucrează doar cu `c`. - **Uiți cazul `n = 0`.** Verifică mereu programul și pentru 0 și pentru un număr de o cifră. - **Inițializezi greșit:** produsul cu 0, minimul cu 0. - **Înmulțești `p` cu 10 la fiecare pas** în loc de doar la cifrele păstrate.

9. La BAC

LA BAC

La subiectul I apar grile cu `%` și `/` („ce valoare are expresia...”). La subiectul al II-lea, aproape în fiecare an, un algoritm în pseudocod care prelucrează cifrele: îl urmărești pe un exemplu, apoi scrii ce face, îl rescrii cu altă structură repetitivă sau găsești valori pentru care afișează un anumit rezultat. La subiectul al III-lea apar subprograme care primesc un număr și întorc un număr construit din cifrele lui.

Exercițiu în stilul subiectului al II-lea. Ce afișează algoritmul pentru `n = 40752`? Ce face în general? (Răspunsul e la finalul lecției.)

```
citește n (număr natural)
x ← 0
┌cât timp n > 0 execută
│   ┌dacă n % 10 > x atunci
│   │   x ← n % 10
│   └■
│   n ← [n / 100]
└■
scrie x
```

10. Exersează pe hârtie

Rezolvă fără calculator. Răspunsurile sunt la finalul lecției, după probleme: încearcă întâi singur.

1. Pentru `n = 30507`, ce valori ia `c`, în ordine, în bucla `while (n > 0) { c = n % 10; n = n / 10; }`?

a) 3 0 5 0 7
b) 7 0 5 0 3
c) 7 5 3
d) 30507

2. Care secvență calculează corect numărul de cifre ale lui `n`, inclusiv pentru `n = 0`?

a) `nr = 0; while (n > 0) { nr++; n /= 10; }`
b) `nr = 0; do { nr++; n /= 10; } while (n > 0);`
c) `nr = 1; while (n > 0) { nr++; n /= 10; }`
d) `nr = 0; while (n >= 0) { nr++; n /= 10; }`

3. Ce valoare are `og` după secvența `og = 0; while (n > 0) { og = og * 10 + n % 10; n /= 10; }`, pentru `n = 1200`?

a) 21
b) 0021
c) 1200
d) 2100

4. Ce afișează secvența, pentru `n = 52384`?

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

a) 35
b) 53
c) 284
d) 482

5. Variabila `n` memorează un număr natural de **exact patru cifre**. Care expresie are ca valoare prima lui cifră?

a) `n % 1000`
b) `n / 1000`
c) `n / 10 % 10`
d) `n % 10`

6. Ce inițializare pentru `mn` face ca programul care caută cifra minimă să dea rezultate **greșite**?

a) `mn = 9`
b) `mn = 10`
c) `mn = n % 10`
d) `mn = 0`

7. Câte numere naturale de exact trei cifre sunt palindroame?

8. Scrie cel mai mic număr de trei cifre pentru care metoda cu `p` care **păstrează cifrele pare** dă 0.

11. Mai departe (pentru cine vrea mai mult)

De ce e atât de rapid? Bucla face un pas pentru fiecare cifră. Un număr până la 10^{18} are cel mult 19 cifre, deci 19 pași. Spunem că algoritmul are complexitatea **O(log n)**: numărul de pași crește cu numărul de cifre, nu cu valoarea numărului.

Numărul de cifre fără buclă. `(int)log10(n) + 1` dă numărul de cifre pentru n ≥ 1. E elegant, dar periculos: lucrează cu numere reale, iar pentru valori foarte mari erorile de rotunjire pot da un rezultat greșit. La BAC și la concursuri, bucla e varianta sigură.

De ce funcționează formula cifrei de control. Orice putere a lui 10 dă restul 1 la împărțirea cu 9 (10 = 9 + 1, 100 = 99 + 1, ...). Deci un număr și suma cifrelor lui dau **același rest** la împărțirea cu 9. Repetând adunarea cifrelor, restul rămâne același, iar la final obținem o cifră între 1 și 9 care are acel rest. Asta e exact `1 + (n - 1) % 9` (pentru n > 0).

Numere prea mari pentru `long long`. Dacă un număr are 100 de cifre, nu încape în niciun tip întreg. Îl citești ca **șir de caractere** (lecția X.10) și fiecare caracter `'0'` ... `'9'` devine cifră prin `s[i] - '0'`.

Unde se folosește în practică. Cifrele de control sunt peste tot: ultima cifră a CNP-ului, IBAN-ul, codurile de bare și numerele de card au o cifră calculată din celelalte, ca să prindă greșelile de tastare. Ca provocare: ultima cifră a CNP-ului se obține înmulțind primele 12 cifre, pe rând, cu cifrele constantei 279146358279, adunând produsele și luând restul împărțirii la 11; dacă restul este 10, cifra de control este 1. Scrie un program care verifică un CNP inventat.

Rezumat

- `n % 10` este ultima cifră, `n / 10` taie ultima cifră; cifrele ies de la dreapta la stânga.
- Șablonul: `while (n > 0) { c = n % 10; …; n = n / 10; }`. Schimbi doar inițializarea și ce faci cu `c`.
- Oglindit: `og = og * 10 + c`. Palindrom: oglinditul este egal cu **copia** numărului.
- Număr nou cu cifrele în ordinea inițială: `rez = rez + c * p; p = p * 10;` doar pentru cifrele păstrate.
- Verifică mereu cazurile `n = 0`, un număr de o cifră și numerele cu zerouri la final.

Probleme de lucru (pbinfo)

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

- [ ] #10 **suma-cifrelor** — ușoară
  Șablonul de bază, exact ca în secțiunea 3.
  https://www.pbinfo.ro/probleme/10/suma-cifrelor
- [ ] #66 **numaruldecifre** — ușoară
  Atenție la n = 0 (are o cifră).
  https://www.pbinfo.ro/probleme/66/numaruldecifre
- [ ] #68 **cifra maxima** — ușoară
  https://www.pbinfo.ro/probleme/68/cifra-maxima
- [ ] #69 **oglindit** — ușoară
  https://www.pbinfo.ro/probleme/69/oglindit
- [ ] #127 **prima-cifra** — ușoară
  Prima cifră a fiecărui număr: bucla cu n > 9.
  https://www.pbinfo.ro/probleme/127/prima-cifra

- [ ] #368 **nrpal** — ușoară
  Palindroame într-un interval: oglinditul într-o funcție de gândire, aplicat fiecărui număr.
  https://www.pbinfo.ro/probleme/368/nrpal

Temă (pbinfo)

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

- [ ] #65 **produscifreimpare** — ușoară
  Ce afișezi dacă nu există cifre impare? Citește enunțul.
  https://www.pbinfo.ro/probleme/65/produscifreimpare
- [ ] #77 **ultimacifrapara** — ușoară
  https://www.pbinfo.ro/probleme/77/ultimacifrapara
- [ ] #107 **aparitii** — ușoară
  https://www.pbinfo.ro/probleme/107/aparitii
- [ ] #108 **aparitii2** — ușoară
  Prima cifră = ce rămâne după ce tai toate celelalte.
  https://www.pbinfo.ro/probleme/108/aparitii2
- [ ] #3932 **stergezerouri** — ușoară
  Cât timp ultima cifră e 0, o tai.
  https://www.pbinfo.ro/probleme/3932/stergezerouri
- [ ] #360 **cifra** — ușoară
  Eliminarea unei cifre, păstrând ordinea celorlalte: metoda cu p.
  https://www.pbinfo.ro/probleme/360/cifra
- [ ] #4932 **numarnou** — ușoară
  Diferențele cifrelor alăturate: ai nevoie de două cifre deodată.
  https://www.pbinfo.ro/probleme/4932/numarnou
- [ ] #340 **control** — ușoară
  https://www.pbinfo.ro/probleme/340/control
- [ ] #444 **afisarecifre** — ușoară
  https://www.pbinfo.ro/probleme/444/afisarecifre
- [ ] #2290 **oglindite** — ușoară
  https://www.pbinfo.ro/probleme/2290/oglindite

Pentru cei care vor mai mult

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

- [ ] #361 **cifra1** — ușoară
  https://www.pbinfo.ro/probleme/361/cifra1
- [ ] #1613 **numere15** — ușoară
  https://www.pbinfo.ro/probleme/1613/numere15
- [ ] #365 **cifre-comune** — ușoară
  https://www.pbinfo.ro/probleme/365/cifre-comune
- [ ] #4934 **semioglindit9** — medie
  Prima jumătate și a doua jumătate a cifrelor: numără întâi cifrele.
  https://www.pbinfo.ro/probleme/4934/semioglindit9
- [ ] #1059 **schimbcifre** — medie
  https://www.pbinfo.ro/probleme/1059/schimbcifre
- [ ] #803 **nrsufix** — dificilă
  Pentru cine vrea o problemă cu adevărat grea.
  https://www.pbinfo.ro/probleme/803/nrsufix

Grile din subiectele de BAC (pbinfo)

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

- [ ] #653 **Exercițiul #653**
  algoritm cu cifre alăturate egale (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=653
- [ ] #654 **Exercițiul #654**
  continuarea grilei 653
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=654
- [ ] #1684 **Exercițiul #1684**
  prelucrarea cifrelor într-un interval (Bac 2016)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1684
- [ ] #1734 **Exercițiul #1734**
  număr construit din cifre (de urmărit pe 64598)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1734
- [ ] #977 **Exercițiul #977**
  trei secvențe echivalente sau nu (Bac 2018)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=977
- [ ] #1208 **Exercițiul #1208**
  algoritm în pseudocod (Bac 2021)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1208

- [ ] #1412 **Exercițiul #1412**
  algoritm în pseudocod (Bac 2023)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1412

- [ ] #920 **Exercițiul #920**
  cifre pare și impare
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=920

Citește și exersează în plus

- Cifrele unui număr https://www.pbinfo.ro/articole/65/cifrele-unui-numar
- Cifra de control a unui număr https://www.pbinfo.ro/articole/13257/cifra-de-control-a-unui-numar
- Python Tutor pentru C++ — lipești codul și vezi pas cu pas cum se schimbă n, c și s
  https://pythontutor.com/cpp.html
- CSES — Digit Queries — problemă de concurs (în engleză): a k-a cifră din șirul 123456789101112...
  https://cses.fi/problemset/task/2431

Răspunsuri

Exercițiul de BAC

Afișează **7**. Pașii: n = 40752, cifra 2 → x = 2, n devine 407; cifra 7 → x = 7, n devine 4; cifra 4 nu e mai mare decât 7, n devine 0. Pentru că `n` se împarte la **100**, se sare peste o cifră de fiecare dată: algoritmul afișează **cea mai mare dintre cifrele de pe pozițiile impare, numărate de la dreapta** (unități, sute, zeci de mii…).

Exersează pe hârtie

1. b) Cifrele ies de la dreapta la stânga, iar zerourile **sunt** cifre: 7, 0, 5, 0, 3.

2. b) `do-while` numără o cifră și pentru 0. Varianta a) dă 0 pentru n = 0; c) numără cu una în plus pentru orice n > 0; d) nu se oprește niciodată, pentru că n ajunge la 0 și rămâne 0, iar 0 ≥ 0 e mereu adevărat.

3. a) Pașii: og = 0, 0, 2, 21. Zerourile de la începutul oglinditului nu contează: 0021 este numărul 21.

4. b) Se păstrează cifrele **impare** (5 și 3), în ordinea din număr, prin metoda cu `p`: 53.

5. b) De exemplu 5274 / 1000 = 5. `n % 1000` dă ultimele trei cifre (274), `n / 10 % 10` dă cifra zecilor (7), `n % 10` ultima cifră (4).

6. d) Nicio cifră nu e mai mică decât 0, deci `mn` ar rămâne 0 pentru orice număr. Celelalte trei variante pornesc de la o valoare mai mare sau egală cu orice cifră a numărului.

7. 90. Un palindrom de trei cifre are forma `aba`: prima cifră `a` poate fi 1…9 (9 variante), cifra din mijloc `b` poate fi 0…9 (10 variante), iar ultima e obligatoriu egală cu prima. 9 · 10 = 90. Verifică cu un program care numără palindroamele de la 100 la 999.

8. 100. Rezultatul este 0 când nu se păstrează nicio cifră diferită de 0: fie toate cifrele sunt impare (111), fie singurele cifre pare sunt zerouri. La 100 se păstrează cele două zerouri, dar ele adaugă 0 · 1 + 0 · 10 = 0, iar 1 e impar și se sare. 100 < 111, deci răspunsul e 100. Capcana: mulți răspund 111, gândindu-se doar la „nicio cifră pară".