INFORMATICĂ · CLASA A X-A · FIȘIERE TEXT · LECȚIA 1

Fișiere text: citire și scriere cu ifstream / ofstream

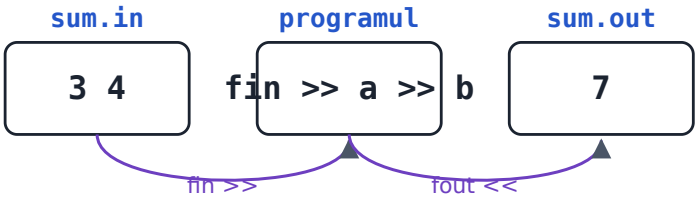
Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-06

CE VEI ȘTI DUPĂ LECȚIE

- Citești date dintr-un fișier text și scrii rezultatele în alt fișier.
- Citești până la sfârșitul fișierului, fără să știi câte valori sunt.
- Rezolvi pe pbinfo problemele care folosesc fișiere (.in / .out).
- Cunoști modelul de citire folosit la Subiectul III.3 de la Bac.

1. De ce fișiere?

Când datele sunt multe (mii, milioane de valori), nu le mai tastăm. Programul le citește dintr-un **fișier text** și scrie rezultatul în alt fișier. Pe pbinfo, la problemele marcate „Fișiere", enunțul îți spune numele exact: de exemplu `sum.in` și `sum.out`.



citești din .in, scrii în .out; nimic nu apare pe ecran

Fluxul de date

Comparație cu ce știai deja:

Ce faci	De la tastatură / ecran	Cu fișiere
citești	`cin >> a;`	`fin >> a;`
scrii	`cout << a;`	`fout << a;`
antet	`<iostream>`	`<fstream>`
de unde vin datele	tastatură, tastate manual	fișierul `.in`, scris o dată

DE CE AȘA?

De ce se folosesc fișiere la probleme? Un test cu 10^6 numere nu se poate tasta. Evaluatorul pune datele într-un fișier, rulează programul și compară fișierul de ieșire cu cel așteptat, automat, pentru zeci de teste în câteva secunde.

2. Șablonul de bază

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

ifstream fin("sum.in");      // fișierul din care citim
ofstream fout("sum.out");    // fișierul în care scriem

int main()
{
    int a, b;
    fin >> a >> b;           // exact ca cin, dar din fișier
    fout << a + b;           // exact ca cout, dar în fișier
    return 0;
}
```

Intrare / ieșire:

Conținut `sum.in`	Conținut `sum.out`
`3 4`	`7`
`-5 2`	`-3`

- `ifstream` = *input file stream* (citire), `ofstream` = *output file stream* (scriere);
- `fin` și `fout` sunt nume alese de noi (poți scrie și `f`, `g`, `in`, `out`);
- fișierele se închid automat la finalul programului (sau explicit cu `fin.close();`).

DE CE AȘA?

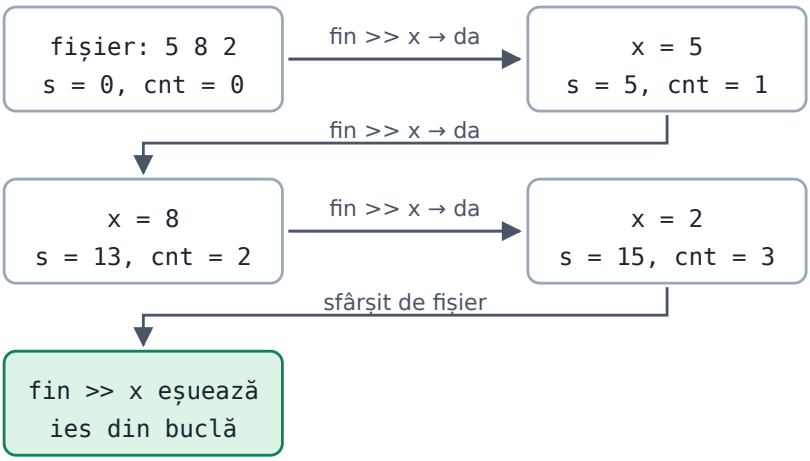
De ce `ifstream fin("sum.in")` e scris în afara lui `main`? Poate fi și înăuntru. Variabilele globale sunt vizibile în toate funcțiile (important când vei scrie subprograme care citesc din același fișier). Pentru programe mici, ambele merg.

CAPCANĂ

Cele mai frecvente motive pentru 0 puncte la problemele cu fișiere: - numele fișierului e scris greșit (`sum.txt` în loc de `sum.in`, majuscule, spații); - ai citit cu `cin` în loc de `fin` sau ai scris cu `cout` în loc de `fout`; - ai uitat `#include <fstream>`.

3. Citirea până la sfârșitul fișierului

Când nu știm câte numere sunt, folosim faptul că `fin >> x` „eșuează” la sfârșitul fișierului:



Citirea până la sfârșit

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

ifstream fin("date.in");
ofstream fout("date.out");

int main()
{
    int x, s = 0, cnt = 0;
    while (fin >> x)          // cât timp s-a putut citi un număr
    {
        s += x;
        cnt++;
    }
    fout << cnt << ' ' << s;
    return 0;
}
```

Urmărire pentru fișierul `5 8 2`:

`fin >> x` reușește?	x	s	cnt
da	5	5	1
da	8	13	2
da	2	15	3
nu (sfârșit) → ies		15	3

DE CE AȘA?

De ce poate fi `fin >> x` pus direct în condiția lui `while`? Expresia `fin >> x` are ca valoare chiar fluxul `fin`, iar un flux în condiție înseamnă „ultima operație a reușit?". Când nu mai e nimic de citit, devine fals și bucla se termină. Citirea și testul se întâmplă într-un singur loc, deci nu poți prelucra o valoare care nu a fost citită.

LA BAC

Acesta este **exact** modelul de citire de la Subiectul III.3 de la Bac: „Fișierul bac.txt conține un șir de cel mult 10^6 numere naturale...". Combinat cu tiparele „din mers" (maxim, secvențe, numărări), îți aduce punctajul de eficiență.

ATENȚIE — GREȘELI FRECVENTE

Nu folosi `while (!fin.eof())`: ultima valoare poate fi prelucrată de două ori dacă fișierul se termină cu un rând gol. Varianta `while (fin >> x)` este cea corectă.

4. Când fișierul are și n pe primul rând

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

ifstream fin("suma.in");
ofstream fout("suma.out");

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

Fișier `suma.in`	Fișier `suma.out`
`3` / `5 2 8`	`15`
`0`	`0`

Când știi `n`, folosește `for`; când nu-l știi, `while (fin >> x)`.

5. Testarea locală (în Code::Blocks sau alt IDE)

1. Creează în **același folder** cu programul un fișier text cu numele cerut (de exemplu `sum.in`).
2. Scrie în el datele din exemplul de pe pbinfo și salvează.
3. Rulează programul, apoi deschide `sum.out` și compară cu ieșirea din exemplu.

SFAT

Dacă vrei să vezi rezultatul pe ecran în timp ce testezi, poți folosi temporar `cout`, dar **nu uita** să revii la `fout` înainte de trimitere.

6. Citirea pe linii (pentru texte)

Când datele sunt propoziții, nu numere:

```
char s[256];
fin.getline(s, 256);     // citește o linie întreagă, cu spații
```

(Detalii la lecția despre șiruri de caractere.)

7. Exemplu tip Bac III.3

„Fișierul `bac.txt` conține cel mult 10^6 numere naturale din intervalul $[0, 10^9]$. Se cere să se afișeze pe ecran cel mai mare număr par din fișier sau mesajul `nu exista`."

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

int main()
{
    ifstream fin("bac.txt");
    int x, mx = -1;
    while (fin >> x)
        if (x % 2 == 0 && x > mx)
            mx = x;
    if (mx == -1) cout << "nu exista";
    else cout << mx;
    return 0;
}
```

Conținutul fișierului	Ieșire	Explicație
`3 8 5 12 7`	`12`	pare: 8 și 12
`3 5 7`	`nu exista`	niciun număr par
`0`	`0`	zero este par

Descrierea eficienței: o singură parcurgere a fișierului, fără memorarea numerelor, doar două variabile.

DE CE AȘA?

De ce `mx = -1` și nu `0`? Numerele sunt naturale, deci `-1` nu poate fi o valoare din fișier: dacă `mx` rămâne `-1`, știm sigur că nu s-a găsit niciun număr par. Cu `0` nu am putea deosebi „nu există par" de „cel mai mare par este 0". Valoarea-santinelă trebuie să fie în afara domeniului datelor.

8. Greșeli frecvente

Greșeala	Efect	Corect
`sum.txt` în loc de `sum.in`	fișier negăsit, 0 puncte	exact numele din enunț
`cin` în loc de `fin`	programul așteaptă tastatura	`fin >> x`
`while (!fin.eof())`	ultima valoare dublată	`while (fin >> x)`
`#include <iostream>` fără `<fstream>`	eroare de compilare	`#include <fstream>`
deschizi `fout` fără să scrii nimic	fișier gol	scrie rezultatul
uiți `fout` și lași `cout`	ieșire în loc greșit	`fout << …`

9. La BAC

> **LA BAC**
>
> Subiectul III.3 citește **întotdeauna** dintr-un fișier (numit `bac.txt` sau `bac.in`) și cere afișare pe ecran sau în alt fișier, după enunț. Citește enunțul cu atenție: dacă cere „pe ecran", folosești `cout`; dacă cere într-un fișier, `fout`.

10. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Fișierul `date.in` conține `4 9 1 6`. Ce conține `date.out` după programul din secțiunea 3?

2. Care dintre secvențe citește corect toate numerele până la sfârșitul fișierului?

a) `while (!fin.eof()) { fin >> x; ... }`
b) `while (fin >> x) { ... }`
c) `for (i = 1; i <= 100; i++) { fin >> x; ... }`
d) `while (fin) { fin >> x; ... }`

3. Ce se întâmplă cu `cnt` dacă `date.in` este un fișier gol?

4. Un program conține `ifstream fin("a.in");` dar în folder există doar `A.IN` (sisteme Linux). Ce observi la rulare?

5. Fișierul conține pe prima linie `n`, apoi `n` perechi `x y`. Scrie bucla care calculează suma lui `x * y`.

6. Scrie programul care citește toate numerele din `numere.in` și scrie în `numere.out` cel mai mare și cel mai mic număr.

7. Fișierul conține `-3 7 0 7 -3`. Ce afișează programul de la secțiunea 7 (cel mai mare număr par)?

11. Mai departe (pentru cine vrea mai mult)

Citire rapidă. La fișiere de milioane de valori, citirea contează. Pune la începutul lui `main`: `ios::sync_with_stdio(false); fin.tie(nullptr);` pentru a accelera `cin` / `fin` de câteva ori. Alternativ, `scanf` / `printf` cu `freopen("sum.in", "r", stdin);` redirecționează tastatura către fișier: nu mai schimbi codul de citire.

Verificarea deschiderii. `if (!fin) { ... }` verifică dacă fișierul s-a deschis. Util în aplicații reale, unde un fișier lipsă nu trebuie să blocheze programul.

Citirea caracter cu caracter. `fin.get(c)` citește și spațiile și sfârșiturile de linie; `fin >> c` le sare. Pentru analiza textelor (numărat cuvinte, litere) alegi `get`.

Date reale. Fișierele CSV (valori separate prin virgulă) sunt, practic, fișiere text. Cu ce ai învățat poți procesa un fișier de la o stație meteo sau un export de note, și aceasta e baza pentru proiecte reale de analiză a datelor.

Rezumat

- `#include <fstream>`; `ifstream fin("nume.in"); ofstream fout("nume.out");`
- Citire până la final: `while (fin >> x) { ... }`.
- Dacă `n` e pe primul rând: `fin >> n;` apoi `for`.
- Verifică de două ori numele fișierelor și că folosești `fin` / `fout`.

Probleme de lucru (pbinfo)

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

- [ ] #1 **sum** — ușoară
  Prima problemă cu fișiere: sum.in → sum.out.
  https://www.pbinfo.ro/probleme/1/sum
- [ ] #57 **n-suma** — ușoară
  https://www.pbinfo.ro/probleme/57/n-suma
- [ ] #7 **maxim2** — ușoară
  https://www.pbinfo.ro/probleme/7/maxim2

Temă (pbinfo)

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

- [ ] #8 **maxim3** — ușoară
  https://www.pbinfo.ro/probleme/8/maxim3
- [ ] #2269 **maximpar** — ușoară
  https://www.pbinfo.ro/probleme/2269/maximpar

- [ ] #2275 **minimdoua** — ușoară
  https://www.pbinfo.ro/probleme/2275/minimdoua
- [ ] #3391 **sirmaxim** — ușoară
  Citire până la sfârșitul fișierului.
  https://www.pbinfo.ro/probleme/3391/sirmaxim
- [ ] #1881 **platou4** — ușoară
  https://www.pbinfo.ro/probleme/1881/platou4

Citește și exersează în plus

- Operații de intrare/ieșire cu fișiere în C++ https://www.pbinfo.ro/articole/19047/operatii-de-intrare-iesire-cu-fisiere-in-cpp
- cppreference — std::ifstream — referința oficială pentru citirea din fișiere (engleză) https://en.cppreference.com/w/cpp/io/basic_ifstream
- Python Tutor pentru C++ — pentru logica buclei; fișierele nu se pot simula acolo, dar tiparul de citire da https://pythontutor.com/cpp.html
- CSES — Weird Algorithm — o problemă simplă cu citire de la tastatură, bună pentru a compara cu fișierele https://cses.fi/problemset/task/1068

Răspunsuri

Exersează pe hârtie

1. `4 20`: patru numere cu suma 4 + 9 + 1 + 6 = 20.

2. b). La a) ultima valoare e prelucrată de două ori dacă fișierul se termină cu un rând gol; c) citește exact 100 de valori; d) are aceeași problemă ca a).

3. `cnt` rămâne `0`: prima citire eșuează, deci corpul nu se execută și se scrie `0 0`.

4. Pe Windows fișierul se deschide (numele nu depind de majuscule); pe Linux nu (sistem care deosebește majusculele), deci `fin` eșuează și variabilele rămân neinițializate sau 0. Pe pbinfo evaluarea e pe Linux: folosește exact numele din enunț.

5.

```
long long s = 0;
fin >> n;
for (int i = 1; i <= n; i++)
{
    fin >> x >> y;
    s += 1LL * x * y;
}
fout << s;
```

6.

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

ifstream fin("numere.in");
ofstream fout("numere.out");

int main()
{
    int x, mx, mn;
    fin >> x;
    mx = mn = x;
    while (fin >> x)
    {
        if (x > mx) mx = x;
        if (x < mn) mn = x;
    }
    fout << mx << ' ' << mn;
    return 0;
}
```

7. `0`. Singurul număr par este 0 (−3 și 7 sunt impare), iar `0 > -1`, deci `mx = 0`.