INFORMATICĂ · CLASA A X-A · TABLOURI UNIDIMENSIONALE · LECȚIA 2

Tablouri unidimensionale (vectori): declarare, citire, parcurgere

Durată: 60 min Nivel: toate profilurile Înainte: lecțiile ix-08, x-01

CE VEI ȘTI DUPĂ LECȚIE

- Declari un vector și accesezi elementele lui prin indice.
- Citești și afișezi un vector, inclusiv în ordine inversă sau doar anumite poziții.
- Aplici pe vectori algoritmii cunoscuți: sumă, numărare, maxim, poziția maximului.
- Construiești un vector nou dintr-unul dat.
- Eviți ieșirea din vector (indici în afara dimensiunii).

1. De ce vectori?

Până acum am prelucrat numerele „din mers", fără să le memorăm. Dar unele probleme cer să revenim la valori: „afișează numerele în ordine inversă", „câte numere sunt mai mari decât media", „sortează". Pentru asta avem nevoie să le **păstrăm** pe toate: într-un **vector** (tablou unidimensional).

Un vector este un șir de variabile de același tip, așezate una după alta, accesate printr-un **indice**. Gândește-te la un rând de dulapuri numerotate: `a[3]` înseamnă „dulapul cu numărul 3".



ele spune „al câtelea"; valoarea e conținutul că

Vectorul a

indice	0	1	2	3	4	5
`a[i]`	(nefolosit)	7	3	12	5	9

DE CE AȘA?

De ce să nu folosim 5 variabile `a1, a2, a3, a4, a5` ? Pentru că nu poți scrie o buclă peste nume de variabile, dar poți peste indici: `a[i]` cu `i = 1..n`. Cu 10^5 numere ai nevoie de buclă; vectorul este singura variantă.

2. Declarare

```
int a[101];          // 101 elemente: a[0], a[1], ..., a[100]
int n;               // câte elemente folosim efectiv
```

- dimensiunea din declarare este **maximul** posibil (din restricții: „n ≤ 100”);
- în C++ indicii încep de la **0**. Mulți preferă să lucreze de la 1 la n (ca în enunțuri), de aceea declară cu 1 element în plus.

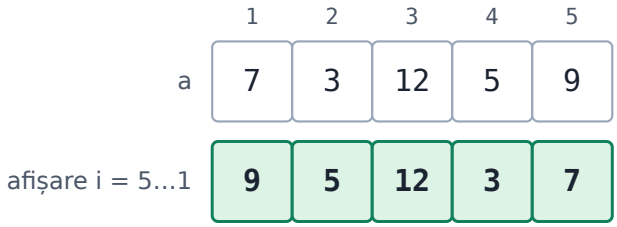
SFAT

Declară vectorii mari **în afara** lui `main` (global). Variabilele globale sunt inițializate automat cu 0 și pot avea dimensiuni mari (de ordinul 10^6–10^7). În `main`, un vector de 10^6 elemente poate „sparge” stiva programului.

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

int a[1001], n;     // global

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = n; i >= 1; i--)      // afișare în ordine inversă
        cout << a[i] << ' ';
    return 0;
}
```



Afișare inversă

Intrare	Ieșire
`5` / `7 3 12 5 9`	`9 5 12 3 7`

DE CE AȘA?

De ce citim *tot* vectorul înainte de afișare? Pentru că primul element afișat este ultimul citit; nu poți ști ce urmează fără să-l fi citit. Aceasta este exact situația în care nu te mai ajută prelucrarea „din mers” și ai nevoie de vector.

3. Algoritmii cunoscuți, pe vectori

Tot ce ai învățat în clasa a IX-a se aplică pe `a[i]` într-o buclă:

```cpp
// suma și numărul elementelor pare
int s = 0, cnt = 0;
for (int i = 1; i <= n; i++)
    if (a[i] % 2 == 0) { s += a[i]; cnt++; }

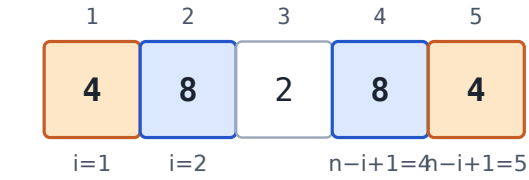
// maximul și poziția lui
int mx = a[1], poz = 1;
for (int i = 2; i <= n; i++)
    if (a[i] > mx) { mx = a[i]; poz = i; }
```

Urmărire pentru `a = (7, 3, 12, 5, 9)`, maximul:

i	a[i]	a[i] > mx ?	mx	poz
1 (init)	7		7	1
2	3	nu	7	1
3	12	da	12	3
4	5	nu	12	3
5	9	nu	12	3

Elementele cu o anumită poziție

```cpp
for (int i = 2; i <= n; i += 2) cout << a[i] << ' ';     // indici pari
for (int i = 1; i <= n / 2; i++)                          // perechi simetrice: a[i] și a[n-i+1]
    if (a[i] == a[n - i + 1]) cnt++;
```



pară cu a[n − i + 1]; elementul din mijloc nu a

Perechi simetrice

LA BAC

Perechea simetrică a poziției `i` într-un vector `1..n` este `n − i + 1`. Pentru indexare de la 0, este `n − 1 − i`. Apare des la verificarea palindromului și la exercițiile de la Bac.

DE CE AȘA?

De ce bucla merge doar până la `n / 2`? Perechile se formează „de la capete spre mijloc”. Dacă mergi până la `n`, fiecare pereche e verificată de două ori (a[1] cu a[n], apoi a[n] cu a[1]), iar la un `cnt` ai număra dublu. Pentru `n` impar, elementul din mijloc nu are pereche.

4. Construirea unui vector nou

Exemplu: în `y` punem elementele pare din `x`, în aceeași ordine.

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

int x[1001], y[1001], n, m;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> x[i];
    for (int i = 1; i <= n; i++)
        if (x[i] % 2 == 0)
            y[++m] = x[i];          // întâi m crește, apoi punem pe poziția m
    for (int i = 1; i <= m; i++)
        cout << y[i] << ' ';
    return 0;
}
```

i	x[i]	par?	m după	y
1	3	nu	0	()
2	4	da	1	(4)
3	7	nu	1	(4)
4	8	da	2	(4, 8)
5	10	da	3	(4, 8, 10)
6	5	nu	3	(4, 8, 10)

DE CE AȘA?

De ce `y[++m]` și nu `y[m++]`? `++m` mărește `m` **înainte** de folosire: primul element ajunge pe poziția 1, iar la final `m` este chiar numărul de elemente. `y[m++]` ar folosi vechea valoare (0) și apoi ar mări-o: elementul ar ajunge pe poziția 0, deci ai pierde convenția „poziții de la 1".

Exemplu: `y[i]` = suma dintre `x[i]` și vecinul următor, cu `x[n]` legat de `x[1]` (circular):

```cpp
for (int i = 1; i <= n; i++)
    y[i] = x[i] + x[i % n + 1];   // pentru i = n, i % n + 1 = 1
```

5. Ieșirea din vector

CAPCANĂ

C++ **nu verifică** indicii. `a[n + 1]` sau `a[-1]` citesc/scriu memorie care nu îți aparține: programul dă rezultate aleatorii sau se oprește cu eroare (pe pbinfo: „Killed by signal 11"). Verifică mereu: buclele merg de la 1 la n? Dimensiunea declarată acoperă restricția maximă?

6. Greșeli frecvente

Greșeala	Efect	Corect
`int a[100]; ... a[100]`	în afara vectorului	declară `a[101]` sau folosește indici ≤ 99
bucla `i <= n` cu `a[i+1]`	`a[n+1]` e în afara datelor	merge până la `n - 1`
vector în `main` de 10^6	blocarea programului	vector global
simetric `n - i` în loc de `n - i + 1`	perechi greșite (indexare de la 1)	`n - i + 1`
`n` declarat dar neinițializat	vector necitit	`cin >> n` înainte

7. La BAC

> **LA BAC**
>
> Subiectul I conține grile cu vectori „parcurgere" (ce valoare are `s`, ce afișează secvența). Subiectul II.3 cere să scrii o secvență pentru un vector (de exemplu „mută ultimul element pe prima poziție"). Reflexul: desenează vectorul cu indici și urmărește bucla pe 2–3 iterații.

8. Exersează pe hârtie

Răspunsurile sunt la final: încearcă întâi singur.

1. Vectorul `v` are elementele `(3, 8, 1, 6, 4)` (indexate de la 1). Ce afișează?

```
for (i = 1; i <= 4; i++)
    if (v[i] < v[i + 1]) cout << v[i + 1] - v[i] << ' ';
```

2. Scrie o secvență care mută ultimul element al vectorului pe prima poziție (permutare circulară la dreapta).

3. Se citesc n numere. Afișează câte sunt strict mai mari decât media lor aritmetică. De ce aici avem neapărat nevoie de vector?

4. Ce se întâmplă dacă scrii `int a[5]; for (i = 0; i <= 5; i++) a[i] = i;`?

5. Ce valori au `m` și `y` după `m = 0; y[m++] = 5; y[++m] = 7;`?

6. Pentru `n = 6`, ce perechi de poziții formează `a[i]` și `a[n - i + 1]`?

7. Ce afișează `for (i = 1; i <= n; i += 2) cout << a[i] << ' ';` pentru `a = (7, 3, 12, 5, 9)`?

8. Scrie secvența care construiește vectorul `y` cu elementele lui `x` mai mari decât 10, în ordine inversă.

9. Mai departe (pentru cine vrea mai mult)

`vector<int>` din biblioteca standard. Are mărime variabilă, `push_back` (adaugă la capăt), `size()`, și verifică indicii cu `at()`. Pentru probleme de concurs, `vector` e preferat; la BAC și la pbinfo se cer

vectorii „clasici”. Exemplu: `vector<int> v; v.push_back(5); cout << v.size();`.

Memoria în spatele unui vector. Elementele sunt așezate **consecutiv** în memorie; `a[i]` se calculează ca adresa lui `a` plus `i · sizeof(int)`. De aceea accesul la `a[i]` e instantaneu, indiferent de `i`. De aceea și ieșirea din vector e periculoasă: citești „vecinii” din memorie.

Un vector pentru fiecare idee. Vectorul de frecvență (lecția 6), sumele parțiale (lecția 7) și vectorul de caracteristici (marcăm în `v[x]` dacă x are o proprietate) sunt variante ale aceleiași idei: **indicele are semnificație**, nu doar poziția.

Complexitate. Accesul la un element e O(1); citirea și parcurgerea sunt O(n). Ștergerea și inserarea la mijloc sunt O(n) (lecția următoare): iată de ce, la structuri cu multe inserări, se folosesc liste înlănțuite (clasa a XI-a).

Rezumat

- `int a[NMAX];` global; folosește indici 1..n.
- Toate tiparele de la clasa a IX-a se aplică pe `a[i]` în buclă.
- Simetricul lui `i`: `n − i + 1`. Construcție: `y[++m] = …`.
- Nu ieși din dimensiunea vectorului.

Probleme de lucru (pbinfo)

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

- [ ] #486 **minmax0** — ușoară
  https://www.pbinfo.ro/probleme/486/minmax0
- [ ] #488 **afisare** — ușoară
  Elementele cu indici pari.
  https://www.pbinfo.ro/probleme/488/afisare
- [ ] #491 **suma2** — ușoară
  https://www.pbinfo.ro/probleme/491/suma2
- [ ] #493 **constr** — ușoară
  Construiești vectorul y din x.
  https://www.pbinfo.ro/probleme/493/constr

Temă (pbinfo)

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

- [ ] #487 **numarare2** — ușoară
  https://www.pbinfo.ro/probleme/487/numarare2
- [ ] #489 **afisare1** — ușoară
  https://www.pbinfo.ro/probleme/489/afisare1

- [ ] #494 **constr1** — ușoară
  https://www.pbinfo.ro/probleme/494/constr1
- [ ] #495 **constr2** — ușoară
  https://www.pbinfo.ro/probleme/495/constr2
- [ ] #553 **pozminmax** — ușoară
  https://www.pbinfo.ro/probleme/553/pozminmax
- [ ] #547 **numarare6** — ușoară
  https://www.pbinfo.ro/probleme/547/numarare6
- [ ] #161 **inlocuire** — ușoară
  https://www.pbinfo.ro/probleme/161/inlocuire
- [ ] #492 **numarare3** — ușoară
  Perechi de elemente: două bucle sau elemente simetrice?
  https://www.pbinfo.ro/probleme/492/numarare3

Grile din subiectele de BAC (pbinfo)

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

- [ ] #1034 **Exercițiul #1034**
  ce valoare afișează un program cu vector inițializat
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1034
- [ ] #1035 **Exercițiul #1035**
  suma elementelor după o condiție, pe un vector dat
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1035
- [ ] #1036 **Exercițiul #1036**
  care secvență afișează corect produsul elementelor pare
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1036
- [ ] #1037 **Exercițiul #1037**
  câte componente satisfac o formulă a[i] = i·i – i
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1037
- [ ] #1039 **Exercițiul #1039**
  ce afișează secvența cu a[i] = 0, apoi modificări
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1039
- [ ] #1040 **Exercițiul #1040**
  prima și ultima poziție a unui element
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1040

- [ ] #415 **Exercițiul #415**
  cea mai mare dintre valorile strict negative (Bac 2009)
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=415

Citește și exersează în plus

- Declararea și parcurgerea tablourilor unidimensionale https://www.pbinfo.ro/articole/5583/declararea-si-parcurgerea-tablourilor-unidimensionale
- Python Tutor pentru C++ — vezi cum arată un vector în memorie și cum se schimbă la fiecare pas https://pythontutor.com/cpp.html
- cppreference — tablouri (arrays) — referința oficială, în engleză https://en.cppreference.com/w/cpp/language/array
- CSES — Distinct Numbers / Increasing Array — probleme cu vectori și limite mari pentru după lecție https://cses.fi/problemset/

Răspunsuri

Exersează pe hârtie

1. `5 5`. Perechi: 3 < 8 (afișează 5); 8 < 1 nu; 1 < 6 (afișează 5); 6 < 4 nu.

2.

```
int ult = v[n];
for (int i = n; i >= 2; i--)
    v[i] = v[i - 1];       // mută toate cu o poziție la dreapta, de la capăt
v[1] = ult;
```

3. Trebuie mai întâi media (suma tuturor numerelor, deci după citirea completă), apoi comparăm **fiecare** număr cu ea; fără vector ar trebui să citim a doua oară.

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

int a[1001], n;

int main()
{
    long long s = 0;
    int cnt = 0;
    cin >> n;
    for (int i = 1; i <= n; i++) { cin >> a[i]; s += a[i]; }
    for (int i = 1; i <= n; i++)
        if ((long long)a[i] * n > s) cnt++;     // a[i] > s/n, fără numere reale
    cout << cnt;
    return 0;
}
```

Comparăm `a[i] * n > s` ca să evităm împărțirea cu zecimale.

4. `a[5]` este în afara vectorului (indicii valizi sunt 0–4): se scrie peste altă variabilă. Comportament nedefinit: rezultate greșite sau oprirea programului.

5. `m = 2`; `y[0] = 5` (cu `m++` s-a folosit 0, apoi `m = 1`), apoi `++m` face `m = 2` și `y[2] = 7`. Poziția 1 a rămas necompletată.

6. `(1, 6)`, `(2, 5)`, `(3, 4)`: formula `a[i]` cu `a[7 − i]`.

7. `7 12 9` (indicii 1, 3, 5).

8.

```
int m = 0;
for (int i = n; i >= 1; i--)
    if (x[i] > 10)
        y[++m] = x[i];
```