FIȘĂ DE LUCRU · CLASA A X-A · FIȘA 2 · ACOPERĂ LECȚIA X.3

Vectori: ștergeri, inserări, verificări

Durată: 90 min Nivel: toate profilurile

CE VEI ȘTI DUPĂ LECȚIE

- Urmărești pe hârtie ștergeri și inserări, inclusiv ce rămâne „în spate" în memorie.
- Alegi direcția corectă a mutărilor și metoda cu doi indici pentru ștergeri multiple.
- Scrii verificări „toate / există / ordonat" care nu se dau peste cap la primul element.

Ce trebuie să știi (o pagină)

Operația	Cod	De reținut
șterge poziția `k`	`for (i = k; i < n; i++) a[i] = a[i + 1]; n--;`	stânga → dreapta, până la `n - 1`
inserezi `x` pe poziția `k`	`for (i = n; i >= k; i--) a[i + 1] = a[i]; a[k] = x; n++;`	dreapta → stânga, loc în plus în vector
șterge tot ce nu respectă `P`	`m = 0; for (i = 1; i <= n; i++) if (P) a[++m] = a[i]; n = m;`	doi indici: `i` citește, `m` scrie; O(n)
inserezi mai multe	construiești `b`: `b[++m] = a[i]; if (...) b[++m] = ...;`	sau pe loc, de la coadă
„toate au `P`"	`ok = true;` caut contraexemplu → `ok = false`	oprire: `i <= n && ok`
„există cu `P`"	`ex = false;` găsesc unul → `ex = true`	**nu** pui niciodată înapoi `false`
ordonat crescător	`if (a[i] > a[i + 1]) ok = false;` pentru `i = 1..n - 1`	strict: `>=`
distincte	`for i, for (j = i + 1; ...)`	`j` pornește de la `i + 1`
duplicate vecine	`if (m == 0 || a[i] != a[m]) a[++m] = a[i];`	pe vector sortat elimină toate dublurile

SFAT

Regula mutărilor: mergi **dinspre locul liber**. La ștergere locul liber e la `k` (deci mergi spre dreapta), la inserare e după `a[n]` (deci pornești de la coadă). Dacă ai dubii, desenează 4 căsuțe.

Nivelul 1 · Prezice

★ citește și prezice

P1. Din `a = (6, 1, 8, 3, 9)` se șterge de **două ori** elementul de pe poziția 2. Ce rămâne? Ce poziții din vectorul **inițial** au dispărut?

P2. Pornind de la `(1, 2, 3)`, se fac pe rând: inserare 9 pe poziția 1; ștergere de pe poziția 3; inserare 0 pe poziția 4. Ce vector obții? (Ultima inserare e permisă: poziția `n + 1` înseamnă „la coadă".)

P3. Vectorul are 7 elemente: `(3, 7, 6, 4, 9, 5, 12)`. Ce valoare are `m` și ce conțin **toate** cele 7 căsuțe ale vectorului după secvență?

```
int m = 0;
for (int i = 1; i <= 7; i++)
    if (a[i] % 3 != 0) a[++m] = a[i];
```

P4. Ce afișează pentru `a = (2, 5, 5, 8)` ? Dar pentru `(2, 5, 6, 8)` ? (`n = 4` ; atenție la valoarea lui `i` după buclă.)

```
bool ok = true; int i;
for (i = 1; i < n && ok; i++)
    if (a[i] >= a[i + 1]) ok = false;
cout << ok << ' ' << i;
```

P5. Pentru vectorul sortat `(1, 1, 2, 2, 2, 5, 7, 7)` (`n = 8`), ce conține `a[1..m]` și cât este `m` ?

```
int m = 0;
for (int i = 1; i <= n; i++)
    if (i == 1 || a[i] != a[i - 1]) a[++m] = a[i];
```

P6. Ce face secvența pentru `a = (4, 7, 1)` , `n = 3` ? Ce s-ar obține dacă bucla ar merge de la `i = 1` la `n` ?

```
for (int i = n; i >= 1; i--) { a[2 * i] = a[i]; a[2 * i - 1] = a[i]; }
n = 2 * n;
```

P7. Zigzag. Secvența verifică dacă `a[1] < a[2] > a[3] < a[4] ...` . Ce afișează pentru fiecare vector din tabel?

```
bool ok = true;
for (int i = 1; i < n; i++)
    if ((i % 2 == 1 && a[i] >= a[i + 1]) || (i % 2 == 0 && a[i] <= a[i + 1])) ok = false;
cout << (ok ? "DA" : "NU");
```

`(1, 3, 2, 4)`	`(1, 3, 2, 2)`	`(5, 1, 6)`	`(2, 9)`	`(7)`

P8. Poziție sau valoare? Vectorul `a = (5, 9, 2, 7, 3, 6, 8)` . Ce rămâne după secvența cu `if (i % 3 != 0) a[++m] = a[i];` ? Dar dacă condiția devine `if (a[i] % 3 != 0)` ?

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Ștergerea care sare peste elemente. Programul trebuie să șteargă toate numerele pare din `(2, 4, 5, 6, 8, 1)`. Ce afișează? Repară-l în două feluri: o dată cu o modificare de un rând, o dată cu metoda cu doi indici.

```
for (int i = 1; i <= n; i++)
    if (a[i] % 2 == 0) {
        for (int j = i; j < n; j++) a[j] = a[j + 1];
        n--;
    }
```

P10. Rânduri amestecate, cu o capcană. Ștergeți toate elementele egale cu **primul** element. Pune rândurile în ordine. Un rând arată corect, dar e greșit: care și de ce? Testează-l pe `(3, 5, 3, 5, 7)`.

```
A)  int x = a[1];
B)  int m = 0;
C)  for (int i = 1; i <= n; i++)
D)      if (a[i] != x) a[++m] = a[i];
E)      if (a[i] != a[1]) a[++m] = a[i];
F)  n = m;
```

P11. Completează golurile. Inserează `x` în vectorul **crescător** `a[1..n]`, astfel încât să rămână crescător. Verifică pe `(2, 5, 9, 12)` cu `x = 7`, `x = 1` și `x = 20`. Câte mutări face în fiecare caz?

```
int i = n;
while (i >= 1 && ________) { a[i + 1] = a[i]; i--; }
a[________] = x;
n++;
```

P12. „Există un număr negativ?" Pentru `(-3, 4, 5)` programul spune `NU`. Găsește greșeala.

```
bool exista = false;
for (int i = 1; i <= n; i++)
    if (a[i] < 0) exista = true;
    else exista = false;
cout << (exista ? "DA" : "NU");
```

P13. „Sunt toate elementele distincte?" Programul spune `NU` chiar și pentru `(1, 2, 3)`. De ce? Repară-l.

```
bool d = true;
for (int i = 1; i < n; i++)
    for (int j = 1; j <= n; j++)
        if (a[i] == a[j]) d = false;
```

P14. Ce greșeală a făcut elevul? Trebuia să insereze 8 pe poziția 2 în `(4, 7, 2, 9)`, dar programul afișează `4 8 7 2`. Unde a dispărut 9? Ce rând lipsește?

Nivelul 3 · Scrie cod

★★ scrie cod

Pentru fiecare program, verifică **toate** testele date înainte să spui că e gata.

P15. Coada la casă. Se citesc `n k x`, apoi numerele de bon ale celor `n` clienți din coadă. Clientul de pe poziția `k` pleacă, apoi clientul cu bonul `x` (VIP) intră primul. Afișează coada.

intrare	ieșire
`5 3 99` / `11 12 13 14 15`	`99 11 12 14 15`
`1 1 7` / `4`	`7`
`3 3 5` / `1 2 3`	`5 1 2`

P16. Curățenie în playlist. Se dau duratele (în secunde) a `n` melodii. Șterge, **într-o singură parcurgere**, melodiile mai scurte de 60 s. Afișează câte au rămas și duratele lor. Teste: `6` / `45 200 59 60 180 30` → `3 200 60 180`; `2` / `10 20` → `0`; `3` / `61 70 80` → `3 61 70 80`.

P17. Temperaturi lipsă. Între oricare două temperaturi consecutive care diferă cu **mai mult de 5** grade, inserează media lor (împărțire întreagă). Teste: `4` / `10 20 22 10` → `10 15 20 22 16 10`; `3` / `1 2 3` → `1 2 3`; `2` / `-10 10` → `-10 0 10`.

P18. Liftul. Se dă lista etajelor la care a oprit liftul. Când liftul „oprește" de mai multe ori la rând la același etaj, păstrează o singură oprire. Pe loc, fără al doilea vector.

intrare	ieșire
`7` / `5 5 3 3 3 8 5`	`5 3 8 5`
`3` / `0 0 0`	`0`
`4` / `1 2 1 2`	`1 2 1 2`

P19. Tăierea pachetului. Se citesc `n k` ($0 \le k \le n$) și `n` cărți. Primele `k` cărți se mută, în aceeași ordine, la sfârșitul pachetului. Teste: `5 2` / `1 2 3 4 5` → `3 4 5 1 2`; `3 3` / `7 8 9` → `7 8 9`; `4 1` / `4 3 2 1` → `3 2 1 4`.

P20. Ștergerea unei bucăți. Se citesc `n p q` și vectorul. Șterge elementele de pe pozițiile `p..q` (inclusiv), mutând fiecare element rămas **o singură dată**. Dacă nu rămâne nimic, afișează `vid`.

intrare	ieșire
`7 3 5` / `1 2 3 4 5 6 7`	`1 2 6 7`
`3 1 3` / `4 5 6`	`vid`
`4 4 4` / `9 8 7 6`	`9 8 7`

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Detectivul. Pe un vector cu 6 elemente s-a rulat `m = 0; for (i = 1; i <= 6; i++) if (a[i] != x) a[++m] = a[i];`. La final `m = 3`, iar cele 6 căsuțe conțin `4 1 7 1 7 9`. Cât era `x`? Ce vector putea fi la început? Câte variante există?

P22. Inserare pe loc. După fiecare element par inserează dublul lui, **fără al doilea vector** și în O(n) (fiecare element se mută cel mult o dată). Test: `(3, 4, 7, 10)` → `(3, 4, 8, 7, 10, 20)`. Indiciu: află întâi lungimea finală.

P23. Contraexemplul. Ana spune: „Dacă vectorul are cel mult o pereche vecină cu `a[i] >= a[i + 1]`, atunci, ștergând un singur element, devine strict crescător." Găsește un vector cu 4 elemente pentru care afirmația e falsă.

P24. Cât costă? Vectorul are `n` elemente: întâi `n/2` numere pare, apoi `n/2` impare. Le ștergi pe cele pare prin ștergeri repetate (cu varianta reparată de la P9). Câte atribuiri `a[j] = a[j + 1]` se fac pentru `n = 8`? Dar pentru `n = 2000`? Câte atribuiri face metoda cu doi indici?

P25. Mini-editor. `editor.in` conține `n`, cele `n` numere, apoi comenzi, câte una pe rând, până la sfârșitul fișierului: `S k` (șterge poziția `k`), `I k x` (inserează `x` pe poziția `k`), `A` (scrie vectorul pe un rând în `editor.out`). Scrie programul și verifică-l pe:

```
5
10 20 30 40 50
S 2
I 1 5
A
I 6 99
S 3
A
```

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] #1452 **stergere-element** — ușoară
  https://www.pbinfo.ro/probleme/1452/stergere-element
- [ ] #158 **inserare** — ușoară
  https://www.pbinfo.ro/probleme/158/inserare
- [ ] #1453 **stergere1** — medie
  https://www.pbinfo.ro/probleme/1453/stergere1
- [ ] #500 **verificareordonat** — ușoară
  https://www.pbinfo.ro/probleme/500/verificareordonat

Temă pentru acasă (pbinfo)

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

- [ ] #159 **inseraredupa** — medie
  https://www.pbinfo.ro/probleme/159/inseraredupa
- [ ] #502 **verifdiferite** — ușoară
  https://www.pbinfo.ro/probleme/502/verifdiferite

- [ ] #1321 **sir-munte** — ușoară
  https://www.pbinfo.ro/probleme/1321/sir-munte
- [ ] #162 **permcirc** — medie
  https://www.pbinfo.ro/probleme/162/permcirc
