

# Algoritmi fundamentali de prelucrare a datelor structurate în tablouri



1. Competențe . . . . .	3
2. Algoritmi de căutare . . . . .	4
3. Algoritmi de sortare . . . . .	11
4. Interclasare . . . . .	19
5. Aplicații . . . . .	22
6. Bibliografie și webografie . . . . .	23



## Competențe generale

- *implementarea algoritmilor într-un limbaj de programare*
- *aplicarea algoritmilor fundamentali în prelucrarea datelor*

## Competențe specifice

- *identificarea necesității structurării datelor în tablouri*
- *prelucrarea datelor structurate*
- *utilizarea unui mediu de programare C++*



### Algoritmi pentru căutarea unui element într-un vector

Pentru căutarea unui element într-un vector se pot folosi:

- algoritmul de căutare într-un vector nesortat;
- algoritmul de căutare într-un vector sortat.

Enunț:

Se dau:

- un vector  $\mathbf{v}$  cu  $n$  elemente numere întregi, fiecare element având cel mult patru cifre și
- o valoare întreagă  $x$  de cel mult patru cifre.

Să se stabilească dacă valoarea dată se găsește sau nu printre elementele vectorului, afișându-se pe ecran mesajul „Există” sau mesajul „Nu există”, după cum elementul căutat se află sau nu printre elementele vectorului.



## a. Algoritmul de căutare într-un vector nesortat (căutare secvențială)

Căutarea elementului se face prin parcurgerea secvențială a vectorului.

Se folosește o variabilă `gasit` (denumită variabilă semafor) de tipul `int`, definită astfel:

$$\text{gasit} = \begin{cases} 1, & \text{dacă } x \text{ există în vector} \\ 0, & \text{în caz contrar} \end{cases}$$



Se folosește metoda reducerii la absurd: înainte de a începe căutarea se presupune că valoarea  $x$  nu se află în vector (se inițializează `gasit` cu valoarea 0).

Se parcurg elementele vectorului, element cu element și se testează dacă un element din vector este egal cu  $x$ : în caz afirmativ variabila `gasit` devine 1.



```
gasit ← 0
pentru i ← 1, n execută
  dacă v[i] = x atunci
    gasit ← 1
```



## b. Algoritmul de căutare într-un vector sortat (căutare binară)

Algoritmul are la bază principiul înjumătățirii repetate a domeniului în care se caută elementul, prin împărțirea vectorului în doi subvectori.

Notații:

- **li** indicele primului element din vector (limita inferioară);
- **ls** indicele ultimului element din vector (limita superioară);
- **mij** indicele elementului din mijlocul vectorului;

$$mij = \frac{li + ls}{2}$$



Se compară valoarea căutată cu valoarea elementului din mijloc  $v[mij]$ ; dacă cele două valori sunt egale înseamnă că s-a găsit elementul; dacă nu sunt egale, vectorul  $v$  va fi împărțit în doi subvectori:

- subvectorul din stânga:  $v[li], v[li+1], \dots, v[mij-1]$ ;
- subvectorul din dreapta:  $v[mij+1], \dots, v[ls-1], v[ls]$ ;

Căutarea va continua doar în acel subvector în care, în mod logic, elementul  $x$  s-ar putea găsi, după cum elementul din mijloc este mai mare sau mai mic decât elementul căutat;

Procedeeul se repetă până când elementul  $x$  s-a găsit sau până când subvectorul în care trebuia să se mai caute nu mai are elemente.



```
gasit ← 0
li ← 1
ls ← n
cât timp li ≤ ls și gasit = 0 atunci
    mij ← (li + ls) / 2
    dacă v[mij] = x atunci
        gasit ← 1
    altfel
        dacă x < v[mij] atunci
            ls ← mij - 1
        altfel
            li ← mij + 1
```



### Algoritmi pentru sortarea unui vector

Pentru sortarea elementelor unui vector se pot folosi algoritmi:

- algoritmul de sortare prin metoda selecției directe;
- algoritmul de sortare prin metoda bulelor.

Enunț:

*Se citește de la tastatură un vector  $v$  cu  $n$  elemente numere întregi, fiecare element având cel mult patru cifre ( $n < 1000$ ).*

*Să se sorteze elementele vectorului în ordine crescătoare și să se afișeze pe ecran vectorul sortat.*



## a. Algoritmul de sortare prin metoda selecției directe



<http://www.youtube.com/watch?v=Ns4TPTC8whw&feature=related>

Prin această metodă se aduce pe prima poziție elementul cu valoarea cea mai mică din cele  $n$  elemente ale vectorului, apoi se aduce pe poziția a doua elementul cu cea mai mică valoare din ultimele  $n-1$  elemente ale vectorului, apoi se aduce pe a treia poziție elementul cu cea mai mică valoare din ultimele  $n-2$  elemente ale vectorului ș.a.d.m.



```
pentru i=1,n-1 execută
  pentru j=i+1,n executa
    dacă v[i]>v[j] atunci
      v[i]↔v[j]
```

S-a notat cu  $v[i] \leftrightarrow v[j]$  interschimbarea valorii  $v[i]$  cu valoarea  $v[j]$ .



## b. Algoritmul de sortare prin metoda bulelor (bubble sort)



[http://www.youtube.com/watch?v=lyZQPjUT5B4&feature=player\\_embedded#!](http://www.youtube.com/watch?v=lyZQPjUT5B4&feature=player_embedded#!)



Prin această metodă se parcurge vectorul și se compară fiecare element cu succesorul său. Dacă cele două elemente nu sunt în ordine, ele se interschimbă.

Vectorul se parcurge de mai multe ori, până când la o parcurgere completă nu se mai execută nicio interschimbare.



Se utilizează variabila `sortat` de tip `int` care se inițializează cu valoarea `1` la începutul parcurgerii (se presupune că vectorul este sortat) și, dacă în timpul parcurgerii se execută o interschimbare, variabilei `sortat` i se atribuie valoarea `0` (vectorul nu este sortat):

$$\text{sortat} = \begin{cases} 1, & \text{când vectorul este sortat} \\ 0, & \text{în caz contrar} \end{cases}$$

Parcurgerea repetată a vectorului se termină atunci când, la sfârșitul unei parcurgeri, variabila `sortat` nu își modifică valoarea (își păstrează valoarea `1`);



```

sortat ← 0
cât timp sortat=0 execută
  sortat ← 1
  pentru i=,n-1 execută
    dacă v[i]>v[i+1] atunci
      v[i]↔v[i+1]
      sortat ← 0

```

S-a notat cu  $v[i] \leftrightarrow v[i+1]$  interschimbarea valorii  $v[i]$  cu valoarea  $v[i+1]$ .

### Algoritm pentru interclasarea a doi vectori

Interclasarea a doi vectori sortați înseamnă reuniunea celor doi vectori într-un al treilea vector care va fi sortat.

Enunț:

*Se citește de la tastatură un vector  $\mathbf{a}$  cu  $m$  elemente numere întregi, sortat crescător și un vector  $\mathbf{b}$  cu  $n$  elemente numere întregi, sortat crescător. Fiecare element al vectorului având cel mult patru cifre ( $n < 1000$ ,  $m < 1000$ ). Se cere să se afișeze pe ecran cele  $m+n$  elemente din cei doi vectori în ordine crescătoare.*



Se compară primul element din vectorul  $\mathbf{a}$  cu primul element din vectorul  $\mathbf{b}$  și cel mai mic dintre ele se pune în vectorul  $\mathbf{c}$ , înaintând cu o poziție în vectorul din care a fost luat elementul.

Procedeul se repetă până când se epuizează unul din vectori. La final se copie la sfârșitul vectorului  $\mathbf{c}$  toate elementele din vectorul rămas neterminat.



$i \leftarrow 1, j \leftarrow 1, k \leftarrow 0$

cât timp  $i \leq m$  și  $j \leq n$  execută

$k \leftarrow k+1$

dacă  $a[i] < b[j]$  atunci

$c[k] \leftarrow a[i]$

$i \leftarrow i+1$

altfel

$c[k] \leftarrow b[j]$

$j \leftarrow j+1$

dacă  $i \leq m$  atunci

cât timp  $i \leq m$  execută

$k \leftarrow k+1; c[k] \leftarrow a[i]; i \leftarrow i+1$

dacă  $j \leq n$  atunci

cât timp  $j \leq n$  execută

$k \leftarrow k+1; c[k] \leftarrow b[j]; j \leftarrow j+1$



### Fișe de lucru

- Aplicații tablouri unidimensionale (vectori)
- Aplicații tablouri bidimensionale (matrici)



## 6. Bibliografie și webografie

1. Miloșescu Mariana, *Informatică. Manual pentru clasa a IX-a*, Editura Didactică și Pedagogică, București, 2004
2. Munteanu Florin, *Programarea calculatoarelor. Manual pentru licee de informatică clasele X-XII*, Editura Didactică și Pedagogică, București, 1994
3. Logofătu Doina, *Bazele programării în C++*, Editura Polirom, Iași, 2006
4. Popescu C., *Culegere de probleme de informatică*, Editura Donaris-Info, Sibiu, 2002
5. Ministerul Educației, Cercetării și Tineretului, Centrul Național pentru Curriculum și Evaluare în Învățământul Preuniversitar, *Proba scrisă la informatică. Examenul de bacalaureat – Variante (1-100)*, București 2008
6. <http://www.youtube.com/watch?v=Ns4TPTC8whw&feature=related>
7. [http://www.youtube.com/watch?v=lyZQPjUT5B4&feature=player\\_embedded#!](http://www.youtube.com/watch?v=lyZQPjUT5B4&feature=player_embedded#!)

