Căutare binară

Se citește un vector cu n componente numere întregi (numerele se presupun ordonate crescător) și o valoare întreagă ("x"). Să se decidă dacă x se găsește sau nu printre numerele citite, iar în caz afirmativ să se tipărească indicele componentei care conține această valoare.

O rezolvare în care x se compară pe rând cu toate cele n componente reperzintă o pierdere de performanță (nu exploatează faptul că cele n valori sunt în secvență crescătoare). Algoritmul care va fi propus este optim și se poate spune că face parte dintre algoritmii "clasici".

Funcția care va fi implementată va decide dacă valoarea căutată se găsește printre numerele aflate pe poziții de indice cuprins între i și j (inițial, i=1, j=n). Pentru aceasta, se va proceda astfel:

- dacă x coincide cu valoarea de la mijloc, aflată pe poziția de indice (i+j)/2, se tipărește indicele și se revine din apel (problema a fost rezolvată).
- în caz contrar, dacă mai sunt și alte elemente de analizat (adică i<j, deci nu au fost verificate toate pozițiile necesare), problema se descompune astfel:
- dacă x este mai mic decât valoarea testată (din mijloc), înseamnă că nu se poate afla pe pozițiile din partea dreaptă, întrucât acestea sunt cel puțin mai mari decât valoarea testată. Nr se poate găsi doar printre componentele cu indice între i și (i+j)/2 - 1, caz în care se reapelează funcția cu acești parametri;
- dacă x este mai mare decât valoarea testată (din mijloc), înseamnă că nu se poate afla în stânga; se poate găsi doar printre componentele cu indicele între (i+j)/2 + 1 și j, caz în care se reapelează funcția cu acești parametri.
- dacă nu mai sunt alte elemente de analizat (pentru că i=j și valoarea din mijloc, v[i], nu coincide cu x), se concluzionează că x nu apare în cadrul vectorului.

Această problemă nu mai necesită analiza tuturor subproblemelor în care se descompune, ea se reduce la o singură subproblemă, iar partea de combinare a soluțiilor dispare. În linii mari, această rezolvare este tot de tip "divide et impera".

```
#include <iostream>
using namespace std;
int v[100], n, x;

void caut(int li, int ls)
{
  int m = (li+ls)/2;
  if (x==v[m])
    cout<<”gasit, indice=”<<m;
  else
    if (i<j)
      if  (x<v[m])
        caut(i, m-1);
      else caut(m+1, j);
    else cout<<”nu a fost gasit.”;
}

int main( )
{
    cout<<”n= ”; cin>>n;
    for (int i=1; i<=n; i++)
    {
      cout<<”v[“<<i<<”]=”; cin>>v[i];
    }
    cout<<”x= ”; cin>>x;
    caut (0,n-1);
    return 0;
}
```

Exemplu de căutare binară recursivă:

```
#include<iostream>
using namespace std;
int n,x,v[10],m;
int caut (int li, int ls)
{
    if(li>ld)
        return -1;
    else
        {
            m =(s+d)/2;
            if (x==v[m])
                return m;
            if (x<v[m])
                return caut(s,m-1);
            else
                return caut(m+1,d);
        }
}
int main()
{   int j;
    cout<<"n,x ";
    cin>>n>>x;
    cout<<"dati "<<n<<" elemente (in ordine crescatoare).\n";
    for (int i=0;i<n;i++)
        cin>>v[i];
    j=caut(0,n-1);
    if(j!=-1)
       cout<<"elementul "<<x<<" a fost gasit pe pozitia: "<<j;
    else
        cout<<”Elementul nu se gaseste in vector”;
    return 0;
}
```