Algoritmi de căutare

Fie v un tablou unidimensional de dimensiune n şi k un element de acelaşi tip cu elementele tabloului. (k=cheie de căutare).
Căutarea lui k revine la a găsi dacă există sau nu, un element din v de valoare k.
În funcţie de aranjarea elementelor în tabloul v, există 3 tipuri de căutări:

1. *secvențială – nu se specifică nimic despre elementele tabloului v*
2. *secvențială în tabloul ordonat*
3. *căutare binară (caz în care tabloul este ordonat)*

1. *Căutarea secvențială – nu se specifică nimic despre elementele tabloului v*

Funcția care rezolvă această căutare:

```
int cautare(int v[], int n, int k)
{
 i=1;
 while (i<=n && v[i]!=k) i++;
 if (i<=n) return 1;
   else  return 0;
}
```

Programul caut1.cpp

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
int v[20],n,i,k;
clrscr();
printf("n="); scanf("%d",&n);
printf("introduceti elementele vectorului\n");
for (i=1;i<=n;i++)
  {
   printf("v[%d]=",i);
   scanf("%d",&v[i]);
  }
printf("nr cautat in sir =");
scanf("%d",&k);
i=1;
while (i<=n && v[i]!=k) i++;
if (i<=n) printf("Numarul cautat se afla in sir");
   else  printf("Numarul cautat nu se afla in sir");
getch();
}
```

2. *Căutarea secvențială în tabloul ordonat*

Fie v sortat crescător $v_1 \le v_2 \le ... \le v_n$.

La compararea unui element v_i cu k avem următoarele 3 situații:

a. $v_i = k$, algoritmul se încheie, căutarea cu succes

b. $v_i > k$, algoritmul se încheie, căutarea fără succes

c. $v_i < k$, se continuă căutarea dacă i§n

Funcția care rezolvă această căutare:

```
void cautare()
{
 gata=0;
 gasit=0;
 i=1;
 while (i<=n && !gata)
  {
     if (v[i]==k) {
                    gata=1;
                    gasit=1;
                  }
        else if (v[i]>k) gata=1;
                    else i++;
   }
 if (i>n || !gasit)  printf("Numarul cautat nu se afla in sir");
      else  printf("Numarul cautat se afla in sir");
}
```

Programul caut2.cpp

```
#include <stdio.h>
#include <conio.h>
int v[20];
int n,i,k,gata,gasit,ind,aux;
void sortv()
{
do
  {  ind=0;
     for(i=1;i<=n-1;i++)
           {if (v[i]>v[i+1]) {
                        aux=v[i];
                        v[i]=v[i+1];
                        v[i+1]=aux;
                        ind=1;
                          }
           }
  } while (ind!=0);
}
void afisare()
{
for(i=1;i<=n;i++)
  printf("v[%d]=%d  ",i,v[i]);
printf("\n");
```

```
}

void main(void)
{
clrscr();
printf("n="); scanf("%d",&n);
printf("introduceti elementele vectorului\n");
for (i=1;i<=n;i++)
   {
    printf("v[%d]=",i);
    scanf("%d",&v[i]);
   }
sortv();
printf("vectorul sortat \n");
afisare();
printf("nr cautat in sir =");
scanf("%d",&k);
gata=0;
gasit=0;
i=1;
while (i<=n && !gata)
  {
   if (v[i]==k) {
                  gata=1;
                  gasit=1;
                }
     else if (v[i]>k) gata=1;
                  else i++;
   }
if (i>n || !gasit)  printf("Numarul cautat nu se afla in sir");
   else  printf("Numarul cautat se afla in sir");
getch();
}
```

3. *Căutarea binară (caz în care tabloul este ordonat) se poate realiza recursiv sau iterativ.*

Fie v sortat crescător $v_1 \le v_2 \le ... \le v_n$.

Varianta recursivă presupune compararea cheii cu elementul din mijlocul tabloului, obținându-se următoarele 3 situații:

a. $v_{mij} = k$, stop s-a găsit cheia

b. $v_{mij} < k$, se continuă căutarea în secvența $v_{mij+1}, v_{mij=2}, ..., v_n$

c. $v_{mij} > k$, se continuă căutarea în secvența $v_1, v_2, ..., v_{mij-1}$

Funcția recursivă este următoarea:

```
int cautare(int k,int inf, int sup)
 {
  int mij;
  if (inf<=sup)
```

```
          {
           mij=(inf+sup)/2;
           if (v[mij]==k) return mij;
                 else if (v[mij]>k) cautare(k,inf,mij-1);
                                 else cautare(k,mij+1,sup);
           }
       else return 0;
}
```

Programul caut4.cpp

```
#include <stdio.h>
#include <conio.h>
int v[20];
int n,k,c;
void citire()
 {
  int i;
  printf("elementele sirului in ordine crescatoare\n");
  printf("v[1]=");scanf("%d",&v[1]);
  for(i=2;i<=n;i++)
   do
           {
    printf("v[%d]=",i);scanf("%d",&v[i]);
           } while (v[i]<=v[i-1]);
}
void afisare()
 {
  int i;
  for(i=1;i<=n;i++)
    printf("v[%d]=%d  ",i,v[i]);
  printf("\n");
 }
int cautare(int k,int inf, int sup)
 {
  int mij;
  if (inf<=sup)
          {
    mij=(inf+sup)/2;
    if (v[mij]==k) return mij;
          else if (v[mij]>k) cautare(k,inf,mij-1);
                                  else cautare(k,mij+1,sup);
   }
    else return 0;
 }
void main(void)
 {
  clrscr();
  do
```

```
  {
   printf("n="); scanf("%d",&n);
  }while(n<1||n>20);
  citire();
  printf("vectorul introdus \n");
  afisare();
  printf("nr cautat in sir =");
  scanf("%d",&k);
  c=cautare(k,1,n);
  if (c==0) printf("elementul nu este in sir\n");
                  else printf("elementul cautat este pe pozitia %d\n",c);
  getch();
}
```

Fie v sortat crescător $v_1 \le v_2 \le ... \le v_n$.
Varianta iterativă pentru căutarea binară este:

```
int cautare(int v[], int n, int k)
{
  int mij,inf,sup,ind;
  inf=1;sup=n;
  ind=0;
  do
  {
   mij=(inf+sup)/2;
   if (v[mij]==k) ind=1;
         else if (v[mij]<k) inf=mij+1;
                                  else sup=mij-1;
  }while(inf<=sup && ind==0);
  if (ind==1) return mij;
                  else return 0;
}
```

Programul caut3.cpp

```
#include <stdio.h>
#include <conio.h>
int v[20];
int n,k;
void citire()
{
  int i;
  printf("elementele sirului in ordine crescatoare\n");
  printf("v[1]=");scanf("%d",&v[1]);
  for(i=2;i<=n;i++)
   do
          {
    printf("v[%d]=",i);scanf("%d",&v[i]);
          } while (v[i]<=v[i-1]);
}
```

```
void afisare()
 {
  int i;
  for(i=1;i<=n;i++)
     printf("v[%d]=%d  ",i,v[i]);
  printf("\n");
 }
void cautare()
 {
  int mij,inf,sup,ind;
  inf=1;sup=n;
  ind=0;
  do
  {
   mij=(inf+sup)/2;
   if (v[mij]==k) ind=1;
          else if (v[mij]<k) inf=mij+1;
                                    else sup=mij-1;
  }while(inf<=sup && ind==0);
  if (ind==1) printf("elementul este pe pozitia %d\n",mij);
                  else printf("elementul nu este in sir");
 }
void main(void)
 {
  clrscr();
  do
  {
   printf("n="); scanf("%d",&n);
  }while(n<1||n>20);
  citire();
  printf("vectorul introdus \n");
  afisare();
  printf("nr cautat in sir =");
  scanf("%d",&k);
  cautare();
  getch();
 }
```