Tablouri unidimensionale (vectori)

Definire, declarare, prelucrări elementare

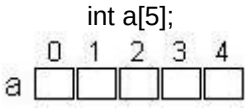
Un **tablou** este o colecţie de date de acelaşi tip, memorate într-o zonă de memorie contiguă, reunite sub un nume comun.
Declararea unei variabile de tip tablou:

tip_dată nume[nr_elemente];

Exemple:

```
int a[5];
float b[10];
char c[5];
```

Observatie: Primul element al vectorului are coordonata 0 (a[0]), al doilea are coordonata 1 (a[1]), s.a.m.d.:



Observaţie: nr_elemente este o constantă întreagă ce specifică numărul de elemente ale vectorului; precizarea valorii sale se poate face şi printr-o constantă simbolică, de exemplu:

```
const int DIM=100;
int a[DIM];
```

Accesul la un element al tabloului se poate face pe baza indicelui acelui element (numerotarea începe de la 0). De exemplu, elementele tabloului c declarat mai sus sunt c[0], c[1], c[2], c[3], c[4].

Un astfel de tablou, pentru care la declarare este specificată o singură dimensiune, iar poziţia unui element este specificată utilizând un singur indice, se numeşte **tablou unidimensional** sau **vector**.

Citirea unui vector se realizează ca în exemplul de mai jos:

```
int a[100],n,i;
cout<<"n="; cin>>n;
for(i=0;i<n; i++)
      {cout<<"a["<<i<<"]=";
      cin>>a[i];
      }
```

Afişarea unui vector:

```
for(i=0;i<n; i++)
      cout<<a[i]<<' ';
```

Determinarea elementului minim/maxim dintr-un vector:

```
int minim=a[0];
for(i=1;i<n; i++)
      if (a[i]<minim)
            minim=a[i];
```

Verificarea unei proprietăţi

Ne întâlnim deseori cu probleme în care trebuie să verificăm dacă toate elementele unui vector au o anumită proprietate, sau dacă există în vector un element care are o anumită proprietate.
Concret, iată cele două situaţii prezentate separat:

a. Se consideră un vector cu n elemente numere naturale (n≤100). Să se verifice dacă toate elementele vectorului sunt numere pare.

Vom folosi o variabilă întreagă, numită `ok`, care va avea valoarea 1 dacă toate elementele vectorului sunt numere pare şi 0 în caz contrar. Presupunem iniţial că toate elementele vectorului sunt numere pare (`ok=1`); parcurgem vectorul şi dacă găsim un element impar, vom atribui variabilei `ok` valoarea 0.

```
unsigned int a[100];
...
ok=1;
for (i=0;i<n &&ok;i++)
        if (a[i]%2!=0)
                ok=0;
```

Se consideră un vector cu n elemente numere întregi (n≤100). Să se verifice dacă există în vector un element negativ.

Vom folosi o variabilă întreagă numită `gasit`, căreia îi vom atribui valoarea iniţială 0 (adică presupunem că toate elementele sunt pozitive). Parcurgem vectorul şi dacă găsim un element negativ, vom atribui variabilei `gasit` valoarea 1.

```
int a[100];
...
gasit=0;
for(i=0;i<n && !gasit; i++)
        if (a[i]<0)
                gasit=1;
```

Căutarea unui element într-un vector

Se consideră un vector cu n componente întregi (n≤100) şi o valoare întreagă x. Să se verifice dacă x apare sau nu în vector.

Observăm că este o situaţie similară celei prezentate anterior la punctul b, adică va trebui să verificăm dacă există în vector un element egal cu valoarea x. Această metodă de căutare, în care testăm succesiv elementele vectorului, se numeşte **căutare secvenţială**.
Frecvent apar probleme de **căutare a unui element într-o mulţime *ordonată.*** În această situaţie, comparăm valoarea x cu elementul din mijlocul vectorului şi avem următoarele posibilităţi:

- x este egal cu elementul din mijloc, deci am terminat căutarea
- x este mai mic decât elementul din mijloc, deci vom continua căutarea în prima jumătate a vectorului
- x este mai mare decât elementul din mijloc, deci vom continua căutarea în a doua jumătate a vectorului

Această metodă de căutare, în care lucrăm prin înjumătăţiri succesive, se numeşte **căutare binară**.

Algoritmul de căutare binară este următorul:

```
#include<iostream>
using namespace std;
int main()
{
 int n,a[50],st,dr,gasit,mij, x;
cout<<"n="; cin>>n;
for(i=0;i<n; i++)
            {cout<<"a["<<i<<"]=";
            cin>>a[i];
            }
cout<<"x="; cin>>x;

        st=0;dr=n-1;gasit=0;

        while (!gasit && st<=dr)
                {mij=(st+dr)/2;
                 if (a[mij]==x)
                        gasit=1;
                 else
                        if (a[mij]>x) dr=mij-1;
                        else st=mij+1;
                }

        if (gasit)
                cout<<x<<" se gaseste pe pozitia "<<mij;
        else
                cout<<x<<" nu se afla in vector";
return 0;
}
```

Verificarea unei date calendaristice

Se citeşte o dată calendaristică. Să se verifice dacă este corectă.
Obs. Se va folosi un vector iniţalizat pentru numărul zilelor din fiecare lună.

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

int an[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};

int main()
{
   int z,l,a;
   cout<<"z="; cin>>z;
   cout<<"l="; cin>>l;
   cout<<"a="; cin>>a;
   if (a<0) cout<<"\n an eronat";
```

```
   else
        {
         if ((a%4==0)&&(a%100!=0)||(a%400==0)) an[2]++;
         if (l<1 || l>12)
        cout<<"\nluna eronata";
        else
        if (z<1 || z>an[l])
                        cout<<"\n zi eronata";
                else
                        cout<<"data corecta";
   }
   return 0;
}
```

Bibliografie

- E. Cerchez, M. Serban - Programarea în limbajul C/C++. Volumul I. Editura Polirom
- E. Cerchez –Informatica, Culegere deprobleme pentru liceu. Editura Polirom