Vectori

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[20];

float b[10];

char c[5];
```

unde 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 a declarat mai sus sunt a[0], a[1], a[2].

Operatii:

1.Citirea unui vector

```
int v[100], n, i;

cout<"n=";  cin>>n;

for(i=0;i<n; i++)

{cout<<"v["<<i<<"]=";   cin>>v[i];}
```

2.Afisarea unui vector

```
for(i=0;i<n; i++)

cout<<v[i]<<' ';
```

3.Determinarea sumei elementelor unui vector

```
int s=0;

for(i=0;i<n; i++)

s=s+v[i];
```

4.Determinarea minimului dintr-un vector

```
int min=v[0];

for(i=1;i<n; i++)

if (v[i]<min) min=v[i];
```

5.Sa se verifice daca toate elementele unui vector sunt numere pozitive:

```
sw=1;

for (i=0;i<n && sw; i++)

if (v[i]<0) sw=0;

if(sw==1)

cout<<"Vectorul are toate elementele pozitive";

else cout<<"Vectorul nu are toate elementele pozitive";
```

6.Cautarea binara

```
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”;
```

7.Sortarea unui vector (metoda selectiei directe)

```
for(i=0; i<n-1; i++)



for(j=i+1; j<n; j++)



if (v[i]>v[j])



{aux=v[i]; v[i]=v[j]; v[j]=aux;}
```

VECTORI DE FRECVENȚĂ

Într-un *vector caracteristic*, pe poziția `i` este stocată o informație despre numărul `i`. De cele mai multe ori, pe poziția `i` se va găsi o valoare booleană (`true` sau `false`) care indică, spre exemplu, dacă `i` se află într-o listă dată de întregi sau nu.

Vectorii de frecvență sunt asemanatori vectorilor caracteristici, singura diferență fiind că pe poziția `i` aceștia reține *numărul de apariții* ale lui `i` (frecvența lui `i`).

Se dau n numere numere naturale cu cel mult două cifre fiecare. Să se determine acele numere care apar o singură dată.

```
#include <iostream>
  using namespace std;
  int n,x,fr[100];
  int main(){
    cin>>n;
  for(int i=0;i<n;i++)
   {cin>>x;
   fr[x]++; }
   for(int i=0;i<100;i++)
      if(fr[i]==1)
      cout<<i<<" ";
          return 0;
   }
```

2. Să se scrie un program care citeşte cel mult 1000000 de numere naturale din intervalul închis [0,9] şi determină cel mai mare număr prim citit şi numărul său de apariţii.

```
#include <iostream>
using namespace std;
 int fr[10];
 int main(){
  int n;
  while( cin>>n)
        fr[n]++;
  if(fr[7]!=0)
{        cout<<7<<" "<<fr[7];
        else
}
  if(fr[5] != 0)
{
        cout<<5<<" "<<fr[5];
        else
}
  if(fr[3] != 0 )
{
        cout<<3<<" "<<fr[3];
        else
}
  if(fr[2] != 0 )
{
```

```
            cout<<2<<" "<<fr[2];
    }
            return 0;

          }
```

3. *Se citesc `n` numere naturale de maxim `3` cifre. Să se afișeze, fără repetiții, în ordine crescătoare, numerele pare ce se regăsesc printre numerele date, iar apoi cele impare, în ordine descrescătoare.*

```
#include <iostream>

using std::cin;
using std::cout;

int n;
bool v[1000];

int main() {
   int x;

   cin >> n;
   for (int i = 0; i < n; i++) {
      cin >> x;
      v[x] = true;
   }

   for (int i = 0; i <= 998; i += 2)
      if (v[i])
         cout << i << ' ';
   cout << '\n';

   for (int i = 999; i >= 1; i -= 2)
      if (v[i])
         cout << i << ' ';
   cout << '\n';
   return 0;
}
```

Se dă un număr natural `n` de maxim `8` cifre. Să se afișeze de câte ori apare fiecare cifră în acesta.

VECTORI-DEPLASARI-INSERARI-STERGERI

1.Scrieți un program care să insereze numărul x pe poziția p în vectorul v

```
#include <iostream>

using namespace std;

int main()
{
    int x,p,v[101],i,n;
    cin>>n;
    for(i=1;i<=n;++i)cin>>v[i];
    cin>>x>>p;
    for(i=n;i>=p;--i)v[i+1]=v[i];
    v[p]=x;
    ++n;
    for(i=1;i<=n;++i)cout<<v[i]<<" ";
    return 0;
}
```

2.Scrieți un program care să elimine toate aparițiile numărului x din vectorul v

```
using namespace std;

int main()
{
    int x,v[101],i,j,n;
    cin>>n;
    for(i=1;i<=n;++i)cin>>v[i];
    cin>>x;
    for(i=1;i<=n;++i)
        if(v[i]==x)
    {
        for(j=i;j<=n-1;++j)v[j]=v[j+1];
        --n;--i;
    }
    for(i=1;i<=n;++i)cout<<v[i]<<" ";
    return 0;
}
```

3.permutarea circulara spre stânga, cu o poziție

```
x=v[1];

for(i=2;i<=n;i++)

    v[i-1]=v[i];

v[n]=x;
```

4.permutarea ciculara spre dreapta cu o poziție

```
x=v[n];
```

```
for(i=n-1;i>=1;i++)
      v[i+1]=v[i];
v[1]=x;
```

5.Afișarea tuturor permutărilor spre stânga ale unui vector cu n elemente

```
for(k=1;k<=n;k++)
  { x=v[1];
    for(i=2;i<=n;i++)
         v[i-1]=v[i];
    v[n]=x;
    for(i=1;i<=n;i++)
        fout<<v[i]<<' ';
}
```

6.Afișarea tuturor permutărilor spre dreapta, ale unui vector cu n elemente

```
for(k=1;k<=n;k++)
  { x=v[n];
    for(i=n-1;i>=1;i--)
         v[i+1]=v[i];
    v[1]=x;
    for(i=1;i<=n;i++)
         fout<<v[i]<<' ';
}
```

7.Inversarea valorilor din vector și afișarea vectorului inversat (Ex. n=4, v=(10,23,41,50) — după inversare v=(50, 41,23,10)

```
for(i=1;i<=n/2;i++)
  {
     aux = v[n-i+1];
     v[n-i+1] = v[i];
     v[i] = aux;
  }
```

VECTORI – OPERATII CU MULTIMI

intersecția a 2 vectori (mulțimi)

```
#include <iostream>

using namespace std;

int main()
{
    unsigned a[101],b[101],c[202],n,m,k=0,i,j,ok;
    cin>>n;
    for(i=1;i<=n;++i)cin>>a[i];
    cin>>m;
    for(i=1;i<=m;++i)cin>>b[i];
    for(i=1;i<=n;++i)
    {
        ok=0;
        for(j=1;j<=m&&ok==0;++j)if(a[i]==b[j])ok=1;
        if(ok==1)c[++k]=a[i];
    }
    for(i=1;i<=k;++i)cout<<c[i]<<" ";
    return 0;
}
```

reuniunea a doi vectori (mulțimi)

```
#include <iostream>

using namespace std;

int main()
{
    unsigned a[101],b[101],c[202],n,m,k,i,j,ok;
    cin>>n;k=n;
    for(i=1;i<=n;++i)cin>>a[i];
    cin>>m;
    for(i=1;i<=m;++i)cin>>b[i];
    for(i=1;i<=n;++i)c[i]=a[i];
    for(i=1;i<=m;++i)
    {
        ok=1;
        for(j=1;j<=n&&ok==1;++j)if(a[j]==b[i])ok=0;
        if(ok==1)c[++k]=b[i];
    }
    for(i=1;i<=k;++i)cout<<c[i]<<" ";
    return 0;
}
```

construire mulțime dintr-un vector, elementele vectorului, repetate doar o dată.

```
#include <iostream>

using namespace std;
```

```
int main()
{
    unsigned v[101],u[101],n,i,j,k=0,ok;
    cin>>n;
    for(i=1;i<=n;++i)cin>>v[i];
    for(i=1;i<=n;++i)
    {
        ok=1;
        for(j=1;j<=k&&ok==1;++j)if(u[j]==v[i])ok=0;
        if(ok==1)u[++k]=v[i];
    }
    for(i=1;i<=k;++i)cout<<u[i]<<" ";
    return 0;
}
```

verifica dacă un vector este o mulțime (există sau nu doi termeni egali între ei)

```
#include <iostream>

using namespace std;

int main()
{
    unsigned v[101],n,i,j,ok=1;
    cin>>n;
    for(i=1;i<=n;++i)cin>>v[i];
    for(i=1;i<=n-1&&ok==1;++i)
        for(j=i+1;j<=n&&ok==1;++j)
        if(v[i]==v[j])ok=0;
    if(ok==1)cout<<"Vectorul este o multime.";
    else cout<<"Vectorul nu este o multime.";
    return 0;
}
```

VECTORI-SORT-INTERCLASARE-CAUTARE

1.Sortare prin selectie directa

```
for(int i = 0; i < N - 1; i++)
    {
        for(int j = i + 1; j < N; j++)
        {
            if(Vector[i] > Vector[j])
            {
                int aux = Vector[i];
                Vector[i] = Vector[j];
                Vector[j] = aux;
            }
        }
    }
```

2.Interclasarea a doi vectori

-vectori ce au elementele stocate in ordine crescatoare /descrescatoare. Scopul acestui algoritm este sa creeze un al treilea vector ce contine elementele din cei doi vectori sortate crescator/descrescator.

1. Declaram un vector C cu un contor k = 0
2. Cat timp se afla elemente in ambii vectori (i <= n si j <= m):

- Comparam elementul A_i cu B_j
- Incrementam k
- Adaugam in C, pe pozitia k, elementul cel mai mic intre A_i cu B_j
- Incrementam indicele corespunzator vectorului din care am facut adaugarea (incrementam i daca elementul Ai a fost mai mic, si in caz contrar, incrementam j)

3. Verificam in care dintre cei doi vectori au mai ramas elemente.

- Daca i <= n atunci inseamna ca mai avem elemente in vectorul A, pe care le luam in ordine si le adaugam la finalul vectorului C.
- Daca j <= m atunci inseamna ca mai avem elemente in vectorul B, pe care le luam in ordine si le adaugam la finalul vectorului C.

```
int A[100], B[100], C[200];
    int n, m, k = 0;

    cout << "Introduceti numarul de elemente corespunzator vectorului A:
"; cin >> n;
    cout << "Introduceti elementele vectorului A: ";
    for(int i = 0; i < n; i++)
        cin >> A[i];

    cout << "Introduceti numarul de elemente corespunzator vectorului B:
"; cin >> m;
    cout << "Introduceti elementele vectorului B: ";
    for(int i = 0; i < m; i++)
        cin >> B[i];

    int i = 0, j = 0;
```

```
while(i < n && j < m)
{
    if(A[i] < B[j])
    {
        C[k] = A[i];
        k++;
        i++;
    }
    else
    {
        C[k] = B[j];
        k++;
        j++;
    }
}

if(i <= n)
{
    for(int p = i; p < n; p++)
    {
        C[k] = A[p];
        k++;
    }
}
if(j <= m)
{
    for(int p = j; p < m; p++)
    {
        C[k] = B[p];
        k++;
    }
}

for(int p = 0; p < k; p++)
    cout << C[p] << " ";
```

3.Cautare binara - este un algoritm folosit pentru a gasi un element intr-o **lista ordonata** (crescatoare sau descrescatoare)

Se foloses ctrei variabile, pe care eu le voi denumii: **s**, **d** si **m**.

- **s** – inceputul intervalului in care cautam
- **m** – mijlocul intervalului
- **d** – sfarsitul intervalului in care cautam

Algoritmul verifica de mai multe ori daca m este egal cu valoarea elementului cautat de noi.

In cazul in care **V[m] == x** (unde x – este elementul cautat) – algoritmul a gasit elementul cautat.

- In caz contrar:
    - daca x este mai mic decat V[m], **d** va devenii **m+1**
    - daca x este mai mare decat V[m], **s** va devenii **m-1**

Algoritmul se repeta atata timp cat **s**<= **d**.

```
int Sol = -1, s = 0, d = N;
    while(s <= d)
    {
        int m = (s+d) / 2;
        if(V[m] == x)
        {
            Sol = m;
            break;
        }
        if(V[m] > x)
            d = m - 1;
        if(V[m] < x)
            s = m + 1; }
```