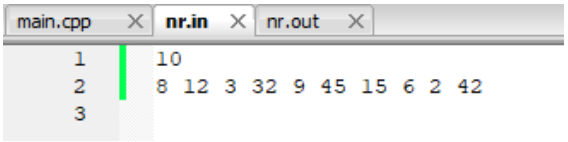
# *Algoritmi de sortare a vectorilor*

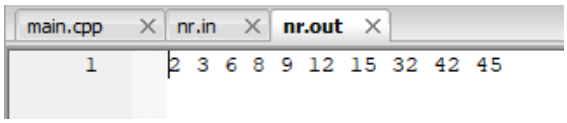
Sortarea unui vector presupune rearanjarea elementelor vectorului astfel încât între valorile lor să existe o relație de ordine (elementele sunt ordonate crescător/descrescător).

1. **Metoda selecției directe** – presupune utilizarea unei variabile auxiliare **aux** pentru a interschimba elementul de pe poziția curentă cu elementul minim găsit. Astfel, 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, ș.a.m.d.

```cpp
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("nr.in");
ofstream g("nr.out");
int n,i,j,v[100],aux;
int main()
{
    f>>n;
    for (i=1;i<=n;i++)
     f>>v[i];
    for (i=1;i<n;i++)
     for (j=i+1;j<=n;j++)
         if (v[i]>v[j])
             {
                 aux=v[i];
                 v[i]=v[j];
                 v[j]=aux;
             }
     for (i=1;i<=n;i++)
         g<<v[i]<<" ";
     return 0;
}
```

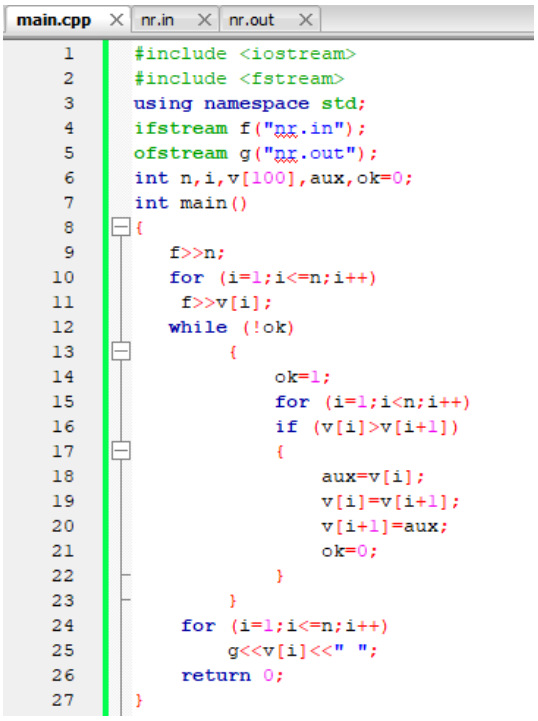


```
10
8 12 3 32 9 45 15 6 2 42
```

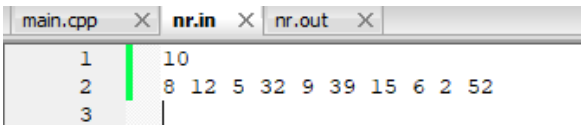


```
2 3 6 8 9 12 15 32 42 45
```

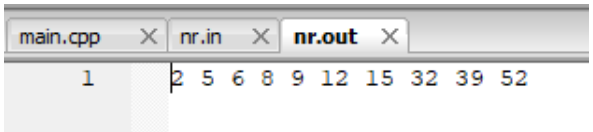
2. **Metoda bulelor** – presupune parcurgerea vectorului și compararea fiecărui element cu succesorul său. Dacă nu sunt în ordine cele două elemente se interschimbă între ele. Vectorul se parcurge până când va fi sortat. Se va utiliza o variabilă **ok**, pe post de variabilă logică care se inițializează cu valoarea **1 (True)** la începutul parcurgerii și, dacă în timpul parcurgerii se execută o interschimbare, variabilei i se atribuie valoarea **0 (False)**. Parcurgerea repetată a vectorului se încheie atunci când, la sfârșitul unei parcurgeri, variabila **ok** nu își modifică valoarea.



```cpp
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("nr.in");
ofstream g("nr.out");
int n,i,v[100],aux,ok=0;
int main()
{
    f>>n;
    for (i=1;i<=n;i++)
     f>>v[i];
    while (!ok)
        {
            ok=1;
            for (i=1;i<n;i++)
            if (v[i]>v[i+1])
            {
                aux=v[i];
                v[i]=v[i+1];
                v[i+1]=aux;
                ok=0;
            }
        }
     for (i=1;i<=n;i++)
         g<<v[i]<<" ";
     return 0;
}
```



```
10
8 12 5 32 9 39 15 6 2 52
```



```
2 5 6 8 9 12 15 32 39 52
```

3. **Metoda inserării directe –** presupune împărțirea vectorului în doi subvectori: sursa (v[i], v[i+1], …, v[n]) și destinația (v[1], v[2], …, v[i]). Elementul v[i] din vectorul sursă este înserat în vectorul destinație în funcție de relația de ordine, astfel încât, vectorul destinație să fie tot timpul ordonat.

main.cpp

```
#include <iostream>
using namespace std;
int i,j,n,aux,v[100];
int main()
{
  cin>>n;
  for (i=0;i<n;i++)
    cin>>v[i];
   for (i=1;i<n;i++)
   {
       aux=v[i];
       j=i-1;
       while (j>0&&aux<v[j])
       {
           v[j+1]=v[j];
           j--;
       }
       if (aux>=v[j])
        v[j+1]=aux;
       else
       {
           v[1]=v[0];
           v[0]=aux;
       }
   }
   for (i=0;i<n;i++)
    cout<<v[i]<<" ";
    return 0;
}
```

```
D:\C++\Sort_ins_dir\bin\Debug\Sort_ins_dir.exe
7
5 2 9 6 12 8 23
2 5 6 8 9 12 23
Process returned 0 (0x0)   execution time : 34.522 s
Press any key to continue.
```

4. **Metoda inserării rapide -** presupune împărțirea vectorului în doi subvectori: sursă și destinație. Deoarece vectorul destinație este un vector ordonat, căutarea poziției în care va fi inserat elementul v[i] se poate realiza cu algoritmul căutării binare.

*main1.cpp

```
#include <iostream>
using namespace std;
int i,j,n,aux,v[100],st,dr,mij;
int main()
{
  cin>>n;
  for (i=0;i<n;i++)
    cin>>v[i];
   for (i=1;i<n;i++)
   {
       aux=v[i]; st=0; dr=i-1;
       while (st<=dr)
       {
           mij=(st+dr)/2;
          if (aux<v[mij])
            dr=mij-1;
          else
            st=mij+1;
       }
       j=i-1;
       while (j>=st)
       {
           v[j+1]=v[j]; j--;
       }
       v[st]=aux;
   }
   for (i=0;i<n;i++)
    cout<<v[i]<<" ";
    return 0;
}
```

```
D:\C++\Sort_ins_dir\bin\Debug\Sort_ins_dir.exe
7
5 2 9 4 1 23 7
1 2 4 5 7 9 23
Process returned 0 (0x0)   execution time : 21.085 s
Press any key to continue.
```

5. **Metoda sortării prin interclasare a doi vectori –** se parcurg simultan cei doi vectori sortați, se compară un element dintr-un vector cu un element din celălalt vector, și elementul cu valoarea cea mai mică (pentru ordonarea crescătoare), este copiat în vectorul destinație. Procesul continuă până când se termină elementele unuia dintre vectori. Elementele rămase în celălalt vector se adaugă la sfârșitul vectorului destinație.

main.cpp | vector.in | vector.out

```cpp
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("vector.in");
ofstream g("vector.out");
int i,j,k,m,n,x[50],y[50],v[100];
int main()
{
    f>>n>>m;
    for (i=0;i<n;i++)
        f>>x[i];
    for (j=0;j<m;j++)
        f>>y[j];
    i=0; j=0; k=0;
    while (i<n && j<m)
    {
        if (x[i]<y[j])
        {
            v[k]=x[i];i++;
        }
        else
        {
            v[k]=y[j];j++;
        }
        k++;
    }
    if (i<n)
        while (i<n)
    {
        v[k]=x[i];
        k++; i++;
    }
    else
        while (j<m)
    {
        v[k]=y[j];
        k++; j++;
    }
    for (k=0;k<n+m;k++)
        g<<v[k]<<" ";
    return 0;
}
```

main.cpp | **vector.in** | vector.out

```
6 10
3 7 9 12 17 34
2 5 14 25 30 33 38 43 56 99
```

main.cpp | vector.in | **vector.out**

```
2 3 5 7 9 12 14 17 25 30 33 34 38 43 56 99
```