Sortarea rapidă (Quick Sort)

Fie tab un tablou cu n componente. Sortarea prin aceasta metoda are la baza urmatorul principiu: practic tabloul este ordonat cand pentru fiecare element din tablou v[poz], elementele situate la stanga sunt mai mici v[poz] iar cele din dreapta sunt mai mari sau egale decat v[poz].

Sortarea tabloului tab decurge astfel:

La un moment dat se prelucreaza o secventa din vector cu indecsi cuprinsi intre p si u

- se ia una din aceste componente, fie ea **piv=v[p],** care se consideră element pivot;
- se fac în tablou interschimbări de componente, astfel încât toate cele mai mici decât valoarea pivot sa treaca în stânga acesteia, iar elementele cu valoare mai mare decât pivot sa treacă în dreapta; prin această operaţie se va deplasa şi valoarea pivot, astfel ca ea nu se va mai gasi pe pozitia initiala ci pe o pozitie corespunzatoare relatiei de ordine. Fie aceasta k. Atentie! In urma acestui set de operatii elementele din stanga sunt mai mici decat pivotul dar nu neaparat si in ordine. La fel cele din dreapta sunt mai mari dar nu neaparat si in ordine

Se continuă setul de operatii similare, aplicând recursiv metoda pentru zona de tablou situată în stânga componentei pivot şi pentru cea din dreapta acesteia;

- - oprirea recursiei se face când lungimea zonei de tablou care trebuie sortată devine egala cu 1.

Fie secventa de vector:
Piv=7
Se compara pivotul cu v[u]. Se interschimba

7	4	2	8	5	9	3	10	1	6

Avansez cu primul: (4<7. Nu se fac interschimbari)

6	4	2	8	5	9	3	10	1	7

Se avanseaza la 2. : (2<7. Nu se fac interschimbari)

6	4	2	8	5	9	3	10	1	7

Se avanseaza cu primul: (8>7. Se interschimba)

6	4	2	8	5	9	3	10	1	7

si se devanseaza in partea dreapta. 7>1.Se interschimba

6	4	2	7	5	9	3	10	1	8

Se avanseaza in stanga. 5<7. Nu se interchimba.

6	4	2	1	5	9	3	10	7	8

Se avanseaza la 9. 9>7 Se interschimba

6	4	2	1	5	9	3	10	7	8

Se devanseaza din dreapta. 7 si 10 sunt in ordine

6	4	2	1	5	7	3	10	9	8

Se trece la 3 care se schimba cu 7.

6	4	2	1	5	7	3	10	9	8

U si p ajung la aceeasi valoare caz in care se incheie secventa.

6	4	2	1	5	3	7	10	9	8

P=u STOP!

K=p
In continuare se aplica aceeasi secventa pe portiunile din stanga respectiv dreapta pivotului:

6	4	2	1	5	3

Si

10	9	8

Etc….
Complexitatea algoritmului QuickSort este **O(n*log(n)).**

Iata o solutie de implementare:

```
/* functia poz rezolva o portiune din vector cuprinsa intre indicii p si u
piv=v[p] este un reper. La sfarsitul executiei acestei functii piv se va gasi pe
o pozitie k cuprinsa intre p si u astfel incat toate componentele vectorului
cuprinse intre p si k-1 vor fi mai mici decat piv iar componentele cuprinse
intre k+1 si u vor fi mai mari decat piv. Deci piv se va gasi pe pozitia k
corespunzatoare ordinii in vector*/

#include <iostream>
using namespace std;
int x[20], n;

int poz(int p,int u)
{int piv,aux,k;
piv=x[p];
while (p<u)
{ if (x[p]>x[u]) {aux=x[p];
                  x[p]=x[u];
                  x[u]=aux;
                  }
if (x[p]==piv)
        u--;
else p++;
}
k=p;
return k;
}

void quick(int p,int u)
{int k;
if (p<u) {k=poz(p,u);
          quick(p,k-1);
          quick(k+1,u);}
}
void main()
{
cout<<"n=";
cin>>n;
for(int i=1;i<=n;i++)
        {cout<<"x["<<i<<"]=";
         cin>>x[i];
        }
quick(0,n-1);
for(i=0;i<n;i++)
cout<<x[i]<<' ';
```

```
return 0;
}
```