

# Sortowanie

Mamy tablicę **t[n]** wypełnioną 10 liczbami

**{5,3,1,11,6,7,4,10,8,9}**

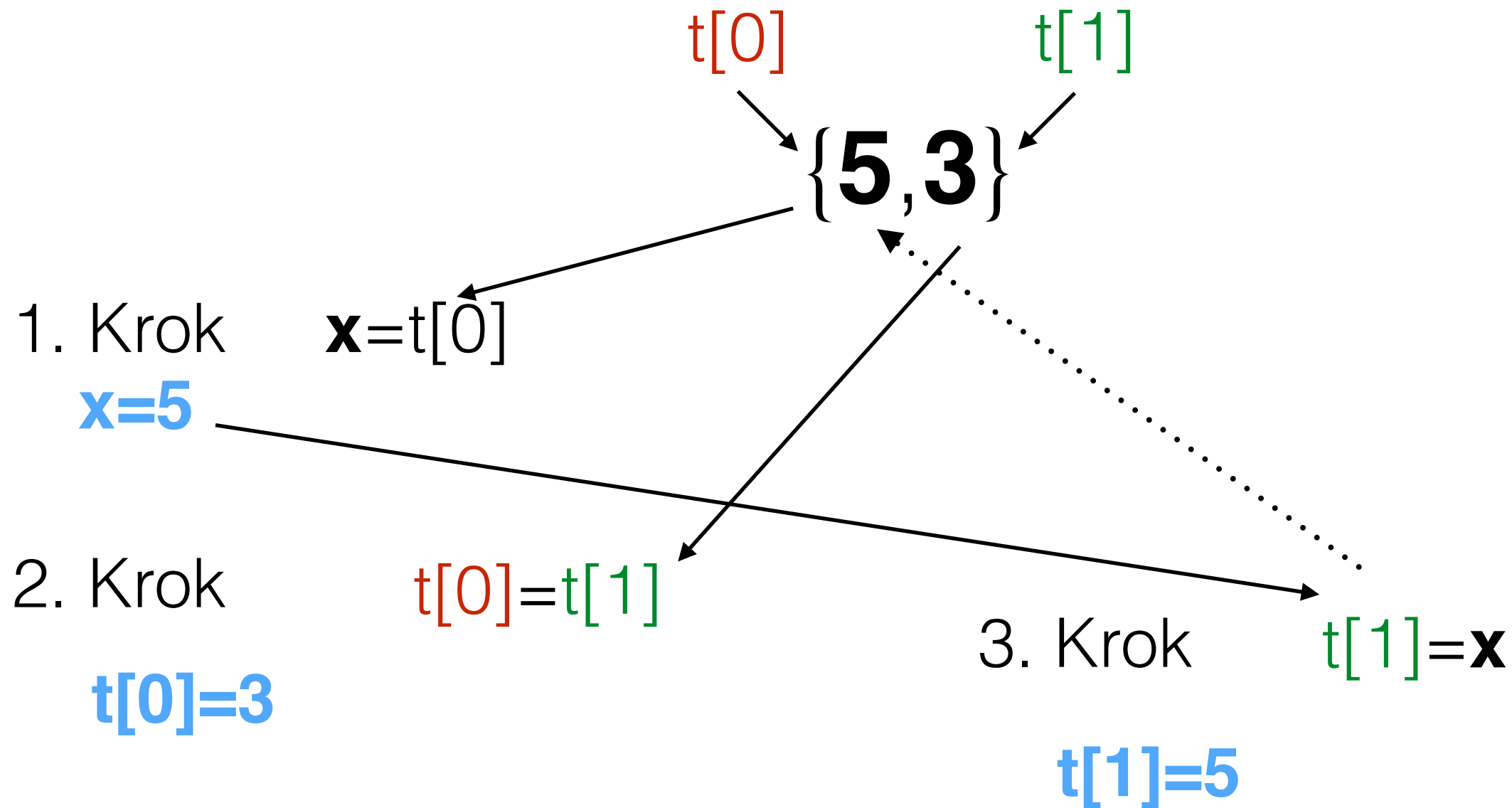
Sortowanie polega na tym, że porównujemy pierwszy element z następnym i jeśli pierwszy jest większy od drugiego to zamieniamy je miejscami.

  
**{5, 3, 1, 11, 6, 7, 4, 10, 8, 9}**

**{3, 5, 1, 11, 6, 7, 4, 10, 8, 9}**

# Zamiana miejsc

Potrzebna będzie zmienna pomocnicza, nazwałem ją **x**



```
x=t[0];  
t[0]=t[1];  
t[1]=x;
```

Następnie porównujemy drugi z trzecim i jeśli drugi jest większy od trzeciego, to też zamieniamy je miejscami

{3, **5**, **1**, 11, 6, 7, 4, 10, 8, 9}



{3, **1**, **5**, 11, 6, 7, 4, 10, 8, 9}

Następnej pary liczb nie zmieniamy, ponieważ trzeci element  
nie jest większy od czwartego

  
{3, 1, **5**, **11**, 6, 7, 4, 10, 8, 9}

{3, 1, **5**, **11**, 6, 7, 4, 10, 8, 9}

Powtarzamy tę operację do końca ciągu liczb.

{3, 1, 5, **11**, **6**, 7, 4, 10, 8, 9}



{3, 1, 5, **6**, **11**, 7, 4, 10, 8, 9}

{3, 1, 5, 6, **11**, **7**, 4, 10, 8, 9}



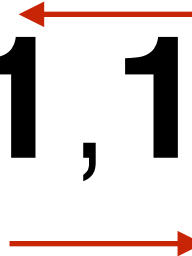
{3, 1, 5, 6, **7**, **11**, 4, 10, 8, 9}

{3, 1, 5, 6, 7, **11**, **4**, 10, 8, 9}



{3, 1, 5, 6, 7, **4**, **11**, 10, 8, 9}

{3, 1, 5, 6, 7, 4, **11**, **10**, 8, 9}



{3, 1, 5, 6, 7, 4, **10**, **11**, 8, 9}

{3, 1, 5, 6, 7, 4, 10, **11**, **8**, 9}



{3, 1, 5, 6, 7, 4, 10, **8**, **11**, 9}

{3, 1, 5, 6, 7, 4, 10, 8, **11**, **9**}





{3, 1, 5, 6, 7, 4, 10, 8, **9**, **11**}



Otrzymujemy taki ciąg liczb

$\{3, 1, 5, 6, 7, 4, 10, 8, \mathbf{9}, \mathbf{11}\}$

**Całą operację powtarzamy od początku**

  
**{3, 1, 5, 6, 7, 4, 10, 8, 9, 11}**  


**{1, 3, 5, 6, 7, 4, 10, 8, 9, 11}**

Uwaga: zamianę miejscami dokonujemy do póty,  
do póki, ciąg liczb nie zostanie uporządkowany.

## Skąd wiemy, że ciąg jest już ułożony poprawnie?

Należy zadeklarować na początku zmienną typu **bool**. Jest to zmienna, która przechowuje dwie informacje: prawdę lub fałsz (**true/false**)

Nazwałem tę zmienną: **zamiana** i przypisałem jej wartość początkową **false**, ponieważ zakładam, że liczby dziwnym trafem są ułożone po kolei. :)

```
bool zamiana=false;
```

Jeśli w trakcie sortowania odbyła się choćby jedna zamiana, to zmieniam wartość zmiennej **zamiana** na **true**

```
for(int i=0;i<9;i++){ // do 9 a nie do 10 bo nie sprawdzam ostatniej z następną, bo jej nie ma.  
    if (t[i]>t[i+1]){ // jeśli badany element jest mniejszy od następnego t[i+1]  
        x=t[i];      // zamiana miejscami dwóch elementów tablicy  
        t[i]=t[i+1];  
        t[i+1]=x;  
        zamiana=true; // ustawiam zmienną na true bo była zamiana  
    }  
}
```

Sortowanie trwa tak  
długo, aż **nie nastąpi**  
ani jedna zamiana

Nie wiem jak długo będzie trwać sortowanie, nie  
wiem ile razy będę powtarzał sprawdzanie czy  
badany element jest większy od następnego.

Wobec tego użyję pętli **do...while**

Gdzie warunkiem trwania pętli będzie informacja, że nastąpiła zamiana elementów

```
do{
```

```
.....
```

```
.....
```

```
}
```

```
while(zamiana==true);
```

Ale tak skonstruowana pętla spowoduje, że sortowanie się nie skończy, program będzie zapętłony.

```
do{  
.....  
.....  
}
```

Do tej pętli należy dodać  
możliwość jej zakończenia

```
while(zamiana==true);
```

Pętla zostanie przerwana, jeśli  
warunek nie będzie spełniony

Zatem na początku pętli ustawiam zmienną **zamiana** na **false**

```
do{  
zamiana=false;  
.....  
}
```

Zakładam, że nie będzie żadnej zamiany,  
co spowoduje wyjście z pętli do...while

```
while(zamiana==true);
```

# Główna część programu

```
do {  
    zamiana=false;    // zakładam, że nie będzie zamiany;  
    for(int i=0;i<9;i++){    // do 9 a nie do 10 bo nie sprawdzam ostatniej z następną, bo jej nie ma.  
        if (t[i]>t[i+1]){    // jeśli badany element jest mniejszy od następnego t[i+1]  
            x=t[i];    // zamiana miejscami dwóch elementów tablicy  
            t[i]=t[i+1];  
            t[i+1]=x;  
            zamiana=true;    // ustawiam zmienna na true bo była zamiana  
        }  
    }  
    for (int i=0;i<10;i++) {    // wydruk kontrolny po każdym zakończeniu pętli for  
        cout <<t[i]<<" ";  
    }  
    cout << endl;    // pusta linia  
} while (zamiana==true);    // do póki była zamiana to pętla trwa
```



```
std::cout << "Hello, World!\nSortowanie liczb!\n";
```

```
for (int i=0;i<10;i++) { //drukowanie nieposortowanej tablicy
```

```
    cout <<t[i]<<" ";
```

```
}
```

```
cout <<endl<<"-----"<< endl;
```

```
do {
```

```
    zamiana=false;    // zakładam, że nie będzie zamiany;
```

```
for(int i=0;i<9;i++){ // do 9 a nie do 10 bo nie sprawdzam ostatniej z następną, bo jej nie ma.
```

```
    if (t[i]>t[i+1]){    // jeśli badany element jest mniejszy od następnego t[i+1]
```

```
        x=t[i];          // zamiana miejscami dwócho elementoów tablicy
```

```
        t[i]=t[i+1];
```

```
        t[i+1]=x;
```

```
        zamiana=true;    // ustawiam zmienna na true bo była zamiana
```

```
    }
```

```
}
```

```
for (int i=0;i<10;i++) {    // wydruk kontrolny po każdym zakończeniu pętli for
```

```
    cout <<t[i]<<" ";
```

```
}
```

```
cout << endl;                // pusta linia
```

```
}
```

```
while (zamiana==true);    // do póki była zamiana to pętla trwa
```

```
// po sortowaniu
```

```
cout<<endl<<endl<<"*-----*"<<endl;
```

```
for (int i=0;i<10;i++) {
```

```
    cout <<t[i]<<" ";
```

```
}
```

```
cout << endl;
```