

**\* Сортировка и поиск информации.  
Методы внутренней сортировки**

Лекция 13

Сортировкой или упорядочением массива называется расположение его элементов по возрастанию (или убыванию). Если не все элементы различны, то надо говорить о неубывающем (или невозрастающем) порядке.

Критерии оценки эффективности этих алгоритмов могут включать следующие параметры:

- количество шагов алгоритма, необходимых для упорядочения;
- количество сравнений элементов;
- количество перестановок, выполняемых при сортировке.

# \*Сортировка подсчетом

В ней используются входной массив  $A$  и вспомогательный массив  $B$  для отсортированного массива. В алгоритме следует для каждого элемента входного массива  $A[i]$  подсчитать количество элементов меньших него ( $c1$ ) и количество элементов, равных ему, но стоящих не после него ( $c2$ )

( $c = c1 + c2$ ). Очевидно что число  $c$  и дает нам номер (позицию) элемента  $A[i]$  в отсортированном массиве, поэтому необходимо  $B[c]$  присвоить  $A[i]$ .

```
program sortpodschet;  
type mas=array[1..100] of real;  
var a,b:mas;  
    i,j,n,c:integer;  
begin
```

```
writeln('Vvedite kol elementov v massive');  
readln(n);  
writeln('Vvedite massiv');  
for i:=1 to n do read (a[i]);  
for i:=1 to n do  
begin  
  c:=0;  
  for j:=1 to n do if a[j]<a[i] then c:=c+1;  
  for j:=1 to i do if a[i]=a[j] then c:=c+1;  
  b[c]:=a[i];  
end;  
for i:=1 to n do write (b[i]:8:3);  
readln;  
end.
```

# \*Сортировка посредством выбора

Самый понятный способ сортировки. Если нам необходимо отсортировать массив по возрастанию, то, возможно мы нашли бы самый маленький элемент и поставили его на первое место, затем самый маленький из оставшихся и его на второе место и т. д.

На  $j$ -ом этапе выбирается элемент наименьший среди  $M[j]$ ,  $M[j+1]$ , . . . ,  $M[N]$  и меняется местами с элементом  $M[j]$ . В результате после  $j$ -го этапа все элементы  $M[1]$ ,  $M[2]$ , . . . ,  $M[j]$  будут упорядочены.

```
program sortViborom;  
type mas=array[1..100] of real;  
var a:mas;  
    i,j,n:integer;
```

```
procedure perest(var k,l:real); {Переставляет элементы k и l}
```

```
  var x:real;
```

```
begin
```

```
  x:=k;  k:=l;  l:=x;
```

```
end;
```

```
procedure findmin(nachind:integer; var minind:integer);
```

```
{Ищет в глобальном массиве a начиная с элемента с номером  
nachind минимальный элемент и его индекс сохраняет в  
переменной minind}
```

```
  var i:integer; min:real;
```

```
begin
```

```
  minind:=nachind;  min:=a[nachind];
```

```
  for i:=nachind+1 to n do
```

```
    if min>a[i] then
```

```
begin
```

```
    min:=a[i]; minind:=i;
```

```
end;
```

```
end;
```

```
begin
```

```
writeln('Vvedite kol elementov v massive'); readln(n);
```

```
writeln('Vvedite massiv'); for i:=1 to n do read (a[i]);
```

```
for j:=1 to n-1 do
```

```
    begin
```

```
        findmin(j,i); perest(a[j],a[i]);
```

```
    end;
```

```
for i:=1 to n do
```

```
    write (a[i]:8:3);
```

```
readln; end.
```

# \* Метод "пузырька" или сортировка обменом

Представьте, что массив расположен вертикально. Элементы с большим значением всплывают вверх наподобие больших пузырьков. При первом проходе вдоль массива, начиная проход "снизу", берется первый элемент и поочередно сравнивается с последующими. При этом:

- если встречается более "легкий" (с меньшим значением) элемент, то они меняются местами;
- при встрече с более "тяжелым" элементом, последний становится "эталонном" для сравнения, и все следующие сравниваются с ним .

В результате наибольший элемент оказывается в самом верху массива.



Во время второго прохода вдоль массива находится второй по величине элемент, который помещается под элементом, найденным при первом проходе, т.е на вторую сверху позицию, и т.д.

```
program puzirkSort;  
type mas=array[1..100] of real;  
var a:mas;  
    i,j,n:integer;  
procedure perest(var k,l:real);  
    var x:real;  
begin  
    x:=k;  
    k:=l;  
    l:=x;  
end;
```

begin

```
writeln('Vvedite kol elementov v massive');
```

```
readln(n);
```

```
writeln('Vvedite massiv');
```

```
for i:=1 to n do
```

```
  read (a[i]);
```

```
for i:=1 to n-1 do
```

```
  for j:=1 to n-i do
```

```
    if a[j]>a[j+1] then perest(a[j],a[j+1]);
```

```
for i:=1 to n do
```

```
  write (a[i]:8:3);
```

```
readln;
```

end.

# \*Сортировка вставками

На  $j$ -ом этапе мы "вставляем"  $j$ -ый элемент  $M[j]$  в нужную позицию среди элементов  $M[1], M[2], \dots, M[j-1]$ , которые уже упорядочены. После этой вставки первые  $j$  элементов массива  $M$  будут упорядочены.

```
program sortVstavkami;  
type mas=array[1..100] of real;  
var a:mas;  i,j,n:integer;  
procedure perest(var k,l:real);  
  var x:real;  
begin  
  x:=k;  k:=l;  
  l:=x;  
end;
```

```
begin
```

```
  writeln('Vvedite kol elementov v massive');  readln(n);
```

```
  writeln('Vvedite massiv');  for i:=1 to n do read (a[i]);
```

```
for j:=2 to n do
```

```
  begin
```

```
    i:=j;
```

```
    while a[i]<a[i-1] do
```

```
      begin
```

```
        perest(a[i],a[i-1]);  i:=i-1;
```

```
        if i=1 then break;
```

```
      end;
```

```
    end;
```

```
  for i:=1 to n do write (a[i]:8:3);
```

```
readln; end.
```

# \* Домашнее задание

1. Составить опорный конспект лекции по теме «Сортировка и поиск информации. Методы внутренней сортировки» на основе презентации.
2. Фундаментальные алгоритмы и структуры данных в Delphi. Бакнелл Джулиан М. СПб.: ООО «ДиаСофтЮП», 2003, стр.153-204.