



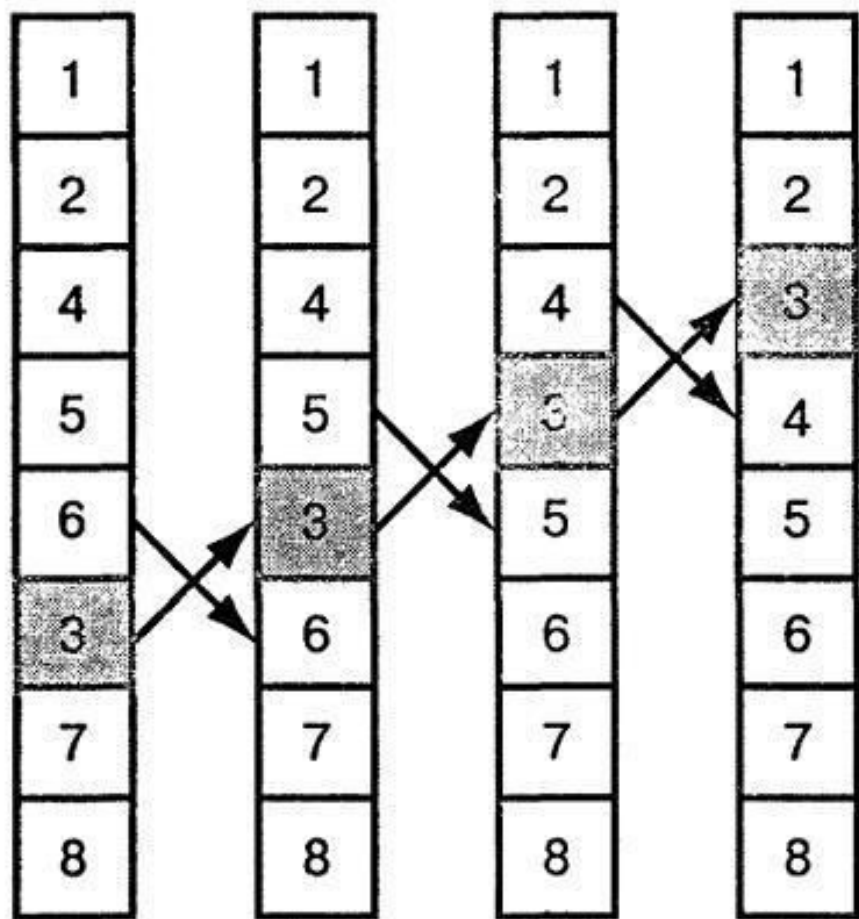
Сортировка прямым обменом

Метод «пузырька»

Алгоритм сортировки прямым обменом основан на принципе сравнения и обмена пары соседних элементов до тех пор, пока не будут отсортированы все элементы.

5	2	1	3	9	0	4	6	8	7
---	---	---	---	---	---	---	---	---	---

Во время каждого прохода сортируемые элементы попарно сравниваются, и если порядок в паре неверный, элементы меняются местами.



Если рассматривать массивы как вертикальные, а не горизонтальные построения, то элементы можно интерпретировать как пузырьки в банке с водой. Элементы с меньшим значением всплывают вверх наподобие пузырьков.

Плюсы:

- Простота реализации алгоритма
- Красивое название(?)

Минусы:

- Один из самых медленных методов сортировки

Алгоритм: Берем элемент массива, сравниваем со следующим, если наш элемент, больше следующего элемента, то мы их меняем местами. После прохождения всего массива, мы можем быть уверены, что максимальный элемент будет "вытолкнут" - и стоять самым последним.

.

Пример первый. Случай убывания:

Дан массив: 0, 5, 8, 4, 9, 3

Расположим элементы списка в процессе убывания.

Т.е. если элемент меньше своего соседа справа — меняется с ним местами.

0, 5, 8, 4, 9, 3

Выполним **первый** проход по массиву:

Сравним 1-й и 2-й: 5, 0, 8, 4, 9, 3

Сравним 2-й и 3-й: 5, 8, 0, 4, 9, 3

Сравним 3-й и 4-й: 5, 8, 4, 0, 9, 3

Сравним 4-й и 5-й: 5, 8, 4, 9, 0, 3

Сравним 5-й и 6-й: 5, 8, 4, 9, 3, 0

Выполним **второй** проход по массиву:

Сравним 1-й и 2-й: **8**, **5**, 4, 9, 3, 0

Сравним 2-й и 3-й: 8, **5**, **4**, 9, 3, 0

Сравним 3-й и 4-й: 8, 5, **9**, **4**, 3, 0

Сравним 4-й и 5-й: 8, 5, 9, **4**, **3**, **0**

Выполним **третий** проход по массиву:

Сравним 1-й и 2-й: **8**, **5**, 9, 4, 3, 0

Сравним 2-й и 3-й: 8, **9**, **5**, 4, 3, 0

Сравним 3-й и 4-й: 8, 9, **5**, **4**, **3**, **0**

Выполним **четвертый** проход по массиву:

Сравним 1-й и 2-й: **9**, **8**, 5, 4, 3, 0

Сравним 2-й и 3-й: 9, **8**, **5**, **4**, **3**, **0**

Выполним **пятый** проход по массиву:

Сравним 1-й и 2-й: **9**, **8**, 5, 4, 3, 0

Результат:

0, 5, 8, 4, 9, 3  **9, 8, 5, 4, 3, 0**

Пример второй. Случай убывания:

Дан массив: 3, 1, 4, 2

Расположим элементы списка в процессе возрастания и напишем программу.

Берем первый элемент "3" сравниваем со следующим "1". Т.к. "3" > "1", то меняем местами: 1 3 4 2

Теперь сравниваем "3" и "4", тройка не больше четвёрки, значит ничего не делаем.

Далее, сравниваем "4" и "2". Четыре больше, чем два - значит меняем местами: 1 3 2 4 .

Цикл закончился. Значит самый большой элемент уже должен стоять на своём месте!!

Видим, что у нас так и произошло. Где бы "4" (наш самый большой элемент) не находился - он всё равно, после прохождения циклом всего массива, будет последним.

Сравниваем "1" и "3" - ничего не меняем.

Сравниваем "3" и "2" - Три больше двух, значит меняем местами. Получается : 1 2 3 4 .

Второй цикл закончили. Мы сделали уже два цикла - значит, с уверенностью можно сказать, что у нас, два последних элемента уже отсортированы. Осталось нам отсортировать третий элемент, а четвёртый, встанет в нужное место, автоматически. Ещё раз, сравниваем первый элемент и второй - видим, что у нас уже всё на своих местах, значит, массив, можно считать, отсортированный по возрастанию элементов.


```
1  const n = 4; {Заводим константу, это будет длина массива}
2  var i, j, k :integer; {Две переменные для вложенного цикла, одна для того чтобы
3  m:array[1..n] of integer; {Заводим массив}
4  begin
5
6  {Будем запрашивать массив с клавиатуры:}
7  Writeln('Введите массив:');
8  for i:=1 to n do begin
9      Writeln(i, ' элемент:');
10     Readln(m[i]);
11 end;
12
13 {Внешний цикл отвечает за то, что мы должны повторить
14  внутренний цикл столько раз, сколько у нас элементов
15  массива минус 1.}
16 for i:=1 to n-1 do begin
17     {Внутренний цикл уже перебирает элементы и сравнивает между собой.}
18     for j:=1 to n-i do begin
19         {Если элемент, больше следующего, то меняем местами.}
20         if m[j]>m[j+1] then begin
21             k:=m[j];
22             m[j]:=m[j+1];
23             m[j+1]:=k;
24         end;
25     end;
26 end;
27
28 {Выводи результат:}
29 for i:=1 to n do
30 Write(m[i], ' ');
31
32
33 end.
```

менять элементы
местами