МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Вятский государственный университет»
(ФГБОУ ВПО «ВятГУ»)

Сортировка обменом (Сортировка пузырьком)

Работу выполнили
студенты группы ИВТ-11:
Веселов Роман
Жуков Дмитрий
Караваев Артем
Макаров Алексей
Пентин Максим
Селиванов Яков
Шеромов Дмитрий

Преимущества и недостатки

* **Сортировка пузырьком** — простейший для понимания и реализации алгоритм сортировки.
* Эффективен он лишь для небольших массивов.
* Недостатком является высокая сложность алгоритма: $O(n^2)$.
* Алгоритм считается учебным и практически не применяется вне учебной литературы, вместо него на практике применяются более эффективные алгоритмы сортировки.
* В то же время метод сортировки обменами лежит в основе некоторых более совершенных алгоритмов, таких как шейкерная сортировка, пирамидальная сортировка и быстрая сортировка.
* Алгоритм является устойчивым (не меняет взаимного расположения равных элементов).
* Алгоритм не использует дополнительной памяти, т.е. все действия осуществляются на одном и том же массиве.

Описание

* Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим наибольшим элементом, а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции как пузырёк в воде, отсюда и название алгоритма).

Пример работы алгоритма

Возьмём массив с числами «5 1 4 2 8» и отсортируем значения по возрастанию, используя сортировку пузырьком. Выделены те элементы, которые сравниваются на данном этапе.

Первый проход:

(**5 1** 4 2 8) (**1 5** 4 2 8)	Здесь алгоритм сравнивает два первых элемента и меняет их местами.
(1 **5 4** 2 8) (1 **4 5** 2 8)	Меняются местами 5 и 4, т. к. 5>4.
(1 4 **5 2** 8) (1 4 **2 5** 8)	Меняются местами 5 и 4, т. к. 5>2.
(1 4 2 **5 8**) (1 4 2 **5 8**)	Алгоритм не меняет местами элементы, т. к. 5<8.

Второй проход:

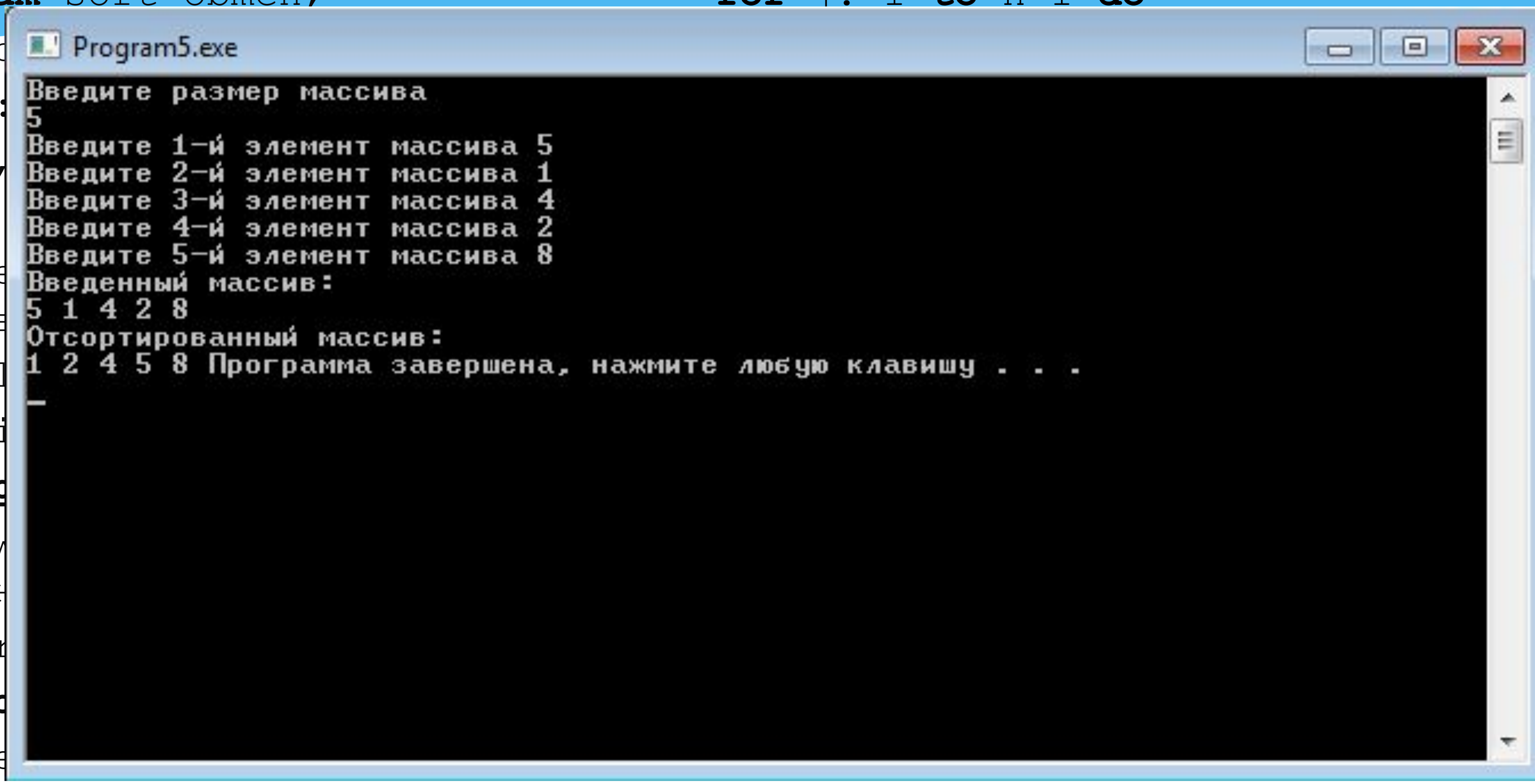
(**1 4** 2 5 8) (**1 4** 2 5 8)	Смена позиций не нужна т. к. 1<4.
(1 **4 2** 5 8) (1 **2 4** 5 8)	Меняются местами, так как 4 > 2.
(1 2 **4 5** 8) (1 2 **4 5** 8)	Смена позиций не нужна т. к. 4<5.
(1 2 4 **5 8**) (1 2 4 **5 8**)	Смена позиций не нужна т. к. 1<4.

Теперь массив отсортирован, но алгоритм не знает так ли это. Поэтому ему необходимо сделать еще один полный проход и определить , что перестановок элементов больше не было.

Код и результат выполнения программы

```
program sort obmen;
uses c
var a:
  n,x,
begin
 write
массив
 readl
 for i
   beg
     w
элемен
     r
   end
 write
 for i:= 1 to n do
   write(a[i],' ');
 writeln;
```

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



Спасибо за внимание!