

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ  
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1  
по курсу «Алгоритмы и структуры данных»  
Тема: Тема работы  
Вариант 11

Выполнил:  
Тимаков Егор(фамилия имя)  
К3141 (номер группы)

Проверила:  
Артамонова В.Е.

Санкт-Петербург  
2023 г.

## **Содержание отчета**

<b>Содержание отчета</b>	<b>2</b>
<b>Задачи по варианту</b>	<b>3</b>
Задача №1. Сортировка вставкой	3
Задача №3. Сортировка вставкой по убыванию	6
Задача №8. Секретарь Своп	11
<b>Вывод</b>	<b>15</b>

## Задачи по варианту

### Задача №1. Сортировка вставкой

Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива  $A = \{31, 41, 59, 26, 41, 58\}$ .

- Формат входного файла (input.txt). В первой строке входного файла содержится число  $n$  ( $1 \leq n \leq 103$ ) — число элементов в массиве. Во второй строке находятся  $n$  различных целых чисел, по модулю не превосходящих 109.

- Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.

- Ограничение по времени. 2сек.

- Ограничение по памяти. 256 мб.

Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

Код:

```
def Solution(Array):
    length = len(Array)
    for i in range(1, length):
        element = Array[i]
        x = i
        while x > 0 and Array[x - 1] > element:
            Array[x] = Array[x - 1]
            x -= 1
        Array[x] = element
    return Array
```

```
InputFile = open("input.txt", "r")

length = int(InputFile.readline())
String = InputFile.readline().split(" ")
InputFile.close()
array = [0] * length
for i in range(0, length):
    array[i] = int(String[i])
```

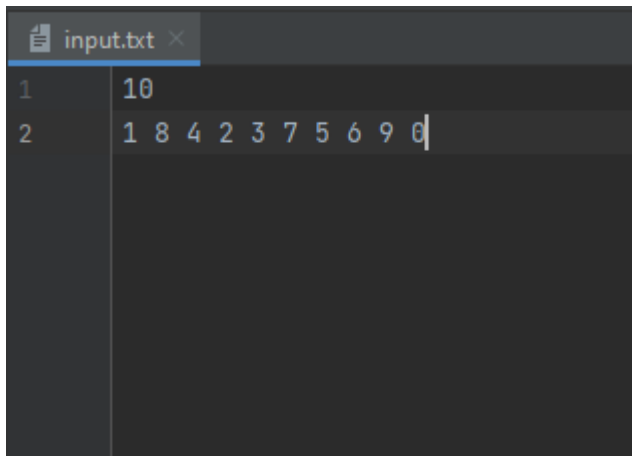
```
array = Solution(array)
OutputFile = open("output.txt", "w")
OutputFile.write(str(array).replace(", ", "")) [1:-1])
OutputFile.close()
```

Объяснение:

Сортировка вставками происходит в функции Solution. Программа меняет каждый элемент массива с предыдущими, пока предыдущие элементы больше текущего.

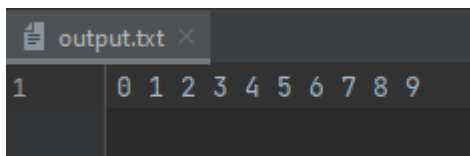
Результат работы программы на примере из задачи:

input.txt:



The screenshot shows a text editor with a file named 'input.txt'. It contains two lines of text: the first line is '10' and the second line is '1 8 4 2 3 7 5 6 9 0'.

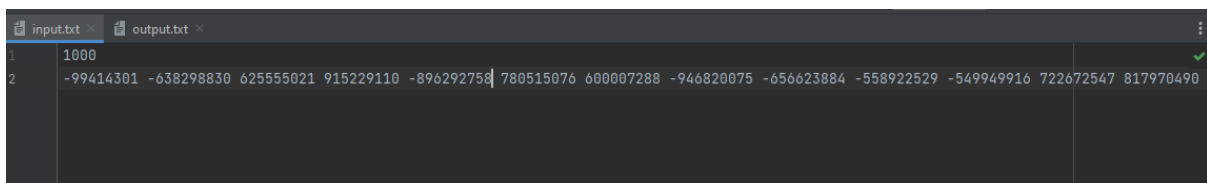
output.txt:



The screenshot shows a text editor with a file named 'output.txt'. It contains a single line of text: '0 1 2 3 4 5 6 7 8 9'.

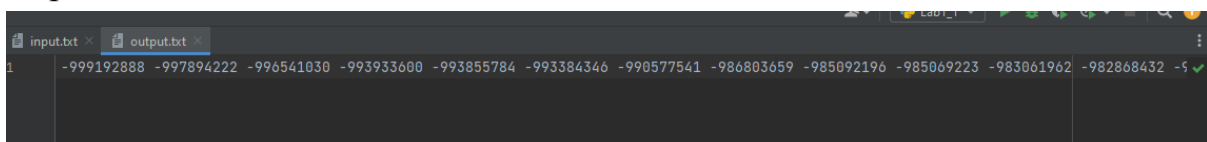
Результат работы кода на максимальных значениях:

input.txt:



The screenshot shows a text editor with two files: 'input.txt' and 'output.txt'. The 'input.txt' file contains two lines of large numbers. The first line is '1000' and the second line is a long string of numbers: '-99414301 -638298830 625555021 915229110 -896292758 780515076 600007288 -946820075 -656623884 -558922529 -549949916 722672547 817970490'.

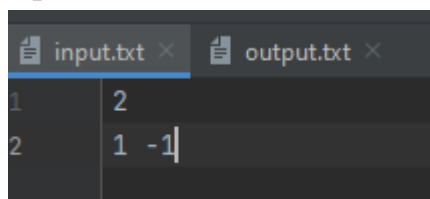
output.txt:



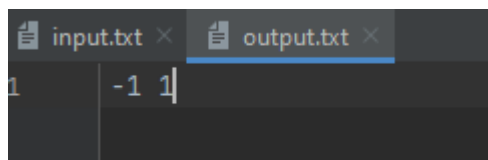
The screenshot shows a text editor with two files: 'input.txt' and 'output.txt'. The 'output.txt' file contains a single line of sorted large numbers: '-999192888 -997894222 -996541030 -993933600 -993855784 -993384346 -990577541 -986803659 -985092196 -985069223 -983061962 -982868432 -982868432'.

Результат работы кода на минимальных значениях:

input.txt:

A screenshot of a text editor showing two tabs: 'input.txt' and 'output.txt'. The 'input.txt' tab is active and contains two lines of text: '1 2' on the first line and '2 1 -1' on the second line. The cursor is at the end of the second line.

output.txt:

A screenshot of a text editor showing two tabs: 'input.txt' and 'output.txt'. The 'output.txt' tab is active and contains one line of text: '-1 1'. The cursor is at the end of the line.

Проверка задачи на время выполнения и затраты памяти

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	20
Пример из задачи	0.003	20
Верхняя граница диапазона значений входных данных из текста задачи	0.03	20

Вывод по задаче:

При решение этой задачи мы освоили алгоритм сортировки вставками. Также мы заметили что время выполнения алгоритма при работе с большими данными увеличивается в 30 раз.

### Задача №3. Сортировка вставкой по убыванию

Текст задачи:

Перепишите процедуру Insertion-sort для сортировки в невозрастающем порядке вместо неубывающего с использованием процедуры Swap. Формат входного и выходного файла и ограничения - как в задаче 1

Код (Без рекурсии):

```
def swap(Array,FirstIndex,SecondIndex):  
    Array[FirstIndex], Array[SecondIndex] = Array[SecondIndex],  
    Array[FirstIndex]
```

```
def Solution(Array):  
    length = len(Array)  
    for i in range(1, length):  
        x = i  
        while x > 0 and Array[x - 1] < Array[x]:  
            swap(Array, x, x-1)  
            x -= 1
```

```
InputFile = open("input.txt", "r")  
Array = [0] * int(InputFile.readline())  
String = InputFile.readline().split(" ")  
InputFile.close()  
for i in range(0,len(Array)):  
    Array[i] = int(String[i])  
Solution(Array)  
OutputFile = open("output.txt", "w")  
OutputFile.write(str(Array).replace(",","")[1:-1])  
OutputFile.close()
```

Объяснение:

Сортировка вставками по убыванию происходит в функции Solution. Программа меняет каждый элемент массива с предыдущими, пока предыдущие элементы меньше текущего.

Результат работы программы на примере из задачи:

input.txt:

```
input.txt x
1 10
2 1 8 4 2 3 7 5 6 9 0
```

output.txt:

```
input.txt x output.txt x
1 9 8 7 6 5 4 3 2 1 0
```

Результат работы кода на максимальных значениях:

input.txt:

```
input.txt x output.txt x
1 1000
2 581047802 -409370498 628990925 160820760 -806029602 -617487920 -692178852 -840862423 813150333 -53333647 651148258 -465861662 -45
```

output.txt:

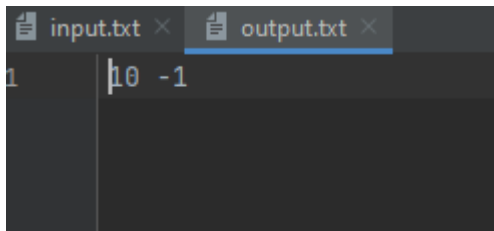
```
input.txt x output.txt x
1 996754772 991960130 990240087 989275594 988676408 984244802 981776533 980439606 979728115 978366915 969098670 968494677 966169
```

Результат работы кода на минимальных значениях:

input.txt

```
input.txt x output.txt x
1 2
2 -1 10
```

output.txt:



Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	20
Пример из задачи	0.0009	20
Верхняя граница диапазона значений входных данных из текста задачи	0.05	20

Код ( C рекурсией):

```
import sys
def swap(Array, FirstIndex, SecondIndex):
    Array[FirstIndex], Array[SecondIndex] = Array[SecondIndex],
    Array[FirstIndex]

def Solution(Array, FirstIndex, SecondIndex):
    i = FirstIndex
    while 0 < i < SecondIndex and Array[i - 1] < Array[i]:
        swap(Array, i, i - 1)
        i -= 1
    if FirstIndex < SecondIndex:
        Solution(Array, FirstIndex + 1, SecondIndex)

sys.setrecursionlimit(10**3+2)
InputFile = open("input.txt", "r")
```



```

Array = [0] * int(InputFile.readline())
String = InputFile.readline().split(" ")
InputFile.close()
for i in range(0, len(Array)):
    Array[i] = int(String[i])
Solution(Array, 1, len(Array))
OutputFile = open("output.txt", "w")
OutputFile.write(str(Array).replace(", ", ""))
OutputFile.close()

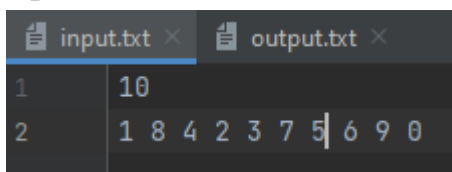
```

Объяснение:

Сортировка вставками по убыванию происходит в функции solution для каждого элемента массива отдельно. Когда текущий элемент массива “оказывается на своем месте” то мы вызываем рекурсию для следующего элемента массива, до тех пор пока функция Solution не достигнет конца массива. Также мы здесь используем функцию swap, которая меняет элементы массива местами. Также мы импортируем библиотеку “sys” для увеличения глубины рекурсии.

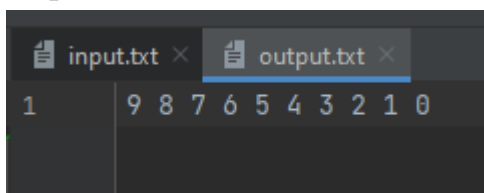
Результат работы программы на примере из задачи:

input.txt:



input.txt	output.txt
1	10
2	1 8 4 2 3 7 5 6 9 0

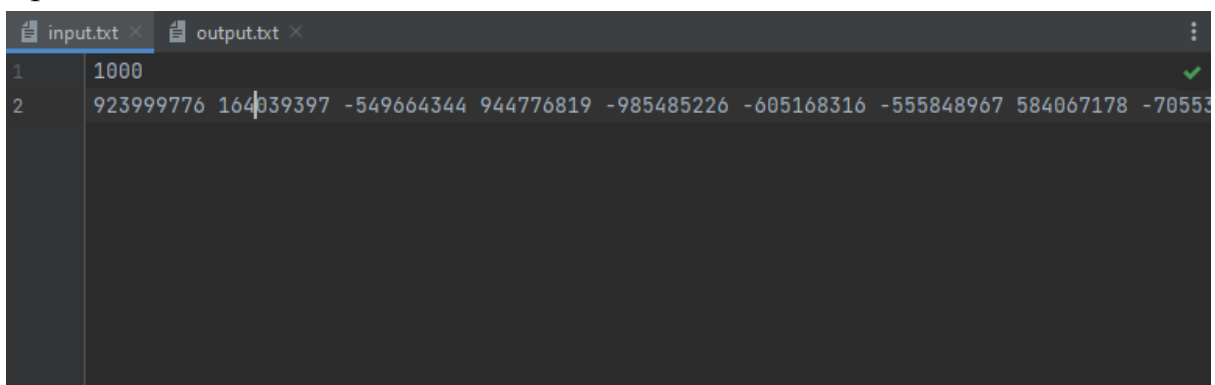
output.txt:



input.txt	output.txt
1	9 8 7 6 5 4 3 2 1 0

Результат работы кода на максимальных значениях:

input.txt:



input.txt	output.txt
1	1000
2	923999776 164939397 -549664344 944776819 -985485226 -605168316 -555848967 584067178 -70553

output.txt:

```
input.txt x output.txt x
1 994390365 993847016 993734499 992064020 990023910 984884774 983585241 981171542 9788198
```

Результат работы кода на минимальных значениях:

input.txt:

```
input.txt x ou
1 2
2 -1 2
```

output.txt:

```
input.txt x output.txt x
1 2 -1
```

Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.0008	20
Пример из задачи	0.0009	20
Верхняя граница диапазона значений входных данных из текста задачи	0.06	20

Вывод по задаче:

В данной задаче мы реализовали алгоритм сортировки по убыванию, без рекурсии и с ее использованием. Мы видим что алгоритм с использованием рекурсии работает быстрее при работе с малыми данными.

## Задача №8. Секретарь Своп

Текст задачи:

Дан массив, состоящий из  $n$  целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

- Формат входного файла (input.txt). В первой строке входного файла содержится число  $n$  ( $3 \leq n \leq 5000$ ) — число элементов в массиве. Во второй строке находятся  $n$  целых чисел, по модулю не превосходящих  $10^9$ . Числа могут совпадать друг с другом.

- Формат выходного файла (output.txt). В первых нескольких строках выведите осуществленные Вами операции перестановки элементов.

Каждая строка должна иметь следующий формат:

Swap elements at indices X and Y.

Здесь  $X$  и  $Y$  — различные индексы массива, элементы на которых нужно переставить ( $1 \leq X, Y \leq n$ ). Мистер Своп любит порядок, поэтому сделайте так, чтобы  $X < Y$ .

После того, как все нужные перестановки выведены, выведите следующую фразу:

No more swaps needed.

Код:

```
def Solution(Array):
    length = len(Array)
    OutputFile = open("output.txt", "w")
    for i in range(1, length):
        x = i
        while x > 0 and Array[x - 1] > Array[x]:
            Array[x] = Array[x - 1]
            x -= 1
```

```

    if(x!=i):
        OutputFile.write("Swap elements at indices " + str(x+1) + " and " +
str(i+1) + ". \n")
        OutputFile.write("No more swaps needed.")

Start_time = datetime.now()
InputFile = open("input.txt", "r")
length = int(InputFile.readline())
String = InputFile.readline().split(" ")
InputFile.close()
array = [0] * length
for i in range(0, length):
    array[i] = int(String[i])
Solution(array)

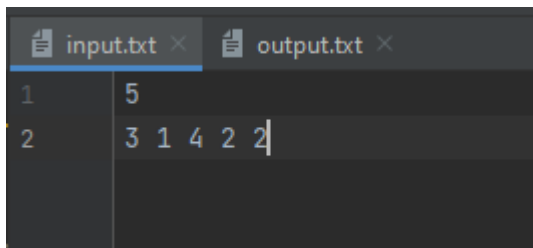
```

Объяснение:

В данной программе мы также реализуем алгоритм сортировки вставкой в функции Solution. Также когда программа меняет элементы массива записывает это в файл output.txt. В конце функция дописывает в файл “No more swaps needed.”

Результат работы программы на примере из задачи:

input.txt:



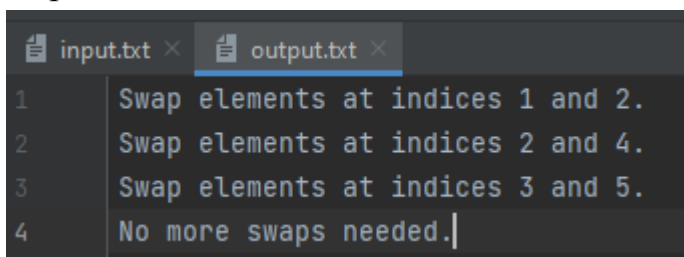
The screenshot shows a text editor with two tabs: 'input.txt' and 'output.txt'. The 'input.txt' tab is active and contains the following text:

```

1 5
2 3 1 4 2 2

```

output.txt:



The screenshot shows a text editor with two tabs: 'input.txt' and 'output.txt'. The 'output.txt' tab is active and contains the following text:

```

1 Swap elements at indices 1 and 2.
2 Swap elements at indices 2 and 4.
3 Swap elements at indices 3 and 5.
4 No more swaps needed.

```

Результат работы кода на максимальных значениях:

input.txt:

input.txt ×		output.txt ×	
1	5000		✓
2	295170907 -995899642 992911891 -567873740 -934411372 601591367 121796981 72685314 -455849		

output.txt:

input.txt ×		output.txt ×	
1	Swap elements at indices 1 and 2.		
2	Swap elements at indices 3 and 4.		
3	Swap elements at indices 4 and 5.		
4	Swap elements at indices 5 and 6.		
5	Swap elements at indices 6 and 7.		
6	Swap elements at indices 7 and 8.		
7	Swap elements at indices 8 and 9.		
8	Swap elements at indices 9 and 10.		
9	Swap elements at indices 10 and 11.		
10	Swap elements at indices 11 and 12.		
11	Swap elements at indices 12 and 13.		
12	Swap elements at indices 13 and 14.		
13	Swap elements at indices 14 and 15.		
14	Swap elements at indices 15 and 16.		
15	Swap elements at indices 16 and 17.		
16	Swap elements at indices 17 and 18.		
17	Swap elements at indices 18 and 19.		
18	Swap elements at indices 19 and 20.		
19	Swap elements at indices 20 and 21.		
20	Swap elements at indices 21 and 22.		
21	Swap elements at indices 22 and 23.		
22	Swap elements at indices 23 and 24.		
23	Swap elements at indices 24 and 25.		
24	Swap elements at indices 25 and 26.		
25	Swap elements at indices 26 and 27.		
26	Swap elements at indices 27 and 28.		

Результат работы кода на минимальных значениях:

input.txt:

```
input.txt x output.txt x
1      3
2     -1 9 2
```

output.txt:

```
input.txt x output.txt x
1      Swap elements at indices 2 and 3.
2      No more swaps needed.
```

Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.0009	20
Пример из задачи	0.001	20
Верхняя граница диапазона значений входных данных из текста задачи	0.009	20

Вывод по задаче:

В этой задаче был реализован алгоритм сортировки вставкой, с демонстрацией элементов, которые поменялись местами.

## **Вывод**

В данной лабораторной мы изучили алгоритм сортировкой вставкой. Также в данной работе мы реализовали данный алгоритм при помощи рекурсии. Вычислительная мощность алгоритма -  $O(n^2)$ . Самый худший набор входных данных для этого алгоритма, когда элементы отсортированы в обратном порядке.