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: Insertion Sort

Выполнил:
Лазарев Марк Олегович
К3241

Санкт-Петербург
2025 г.

Задачи по варианту

Задача №1. Сортировка вставкой

Задача №2. Сортировка выбором

Задача №3. Пузырьковая сортировка

Задачи по варианту

1 задача. 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 ($1 \leq n \leq 10^3$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- **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.

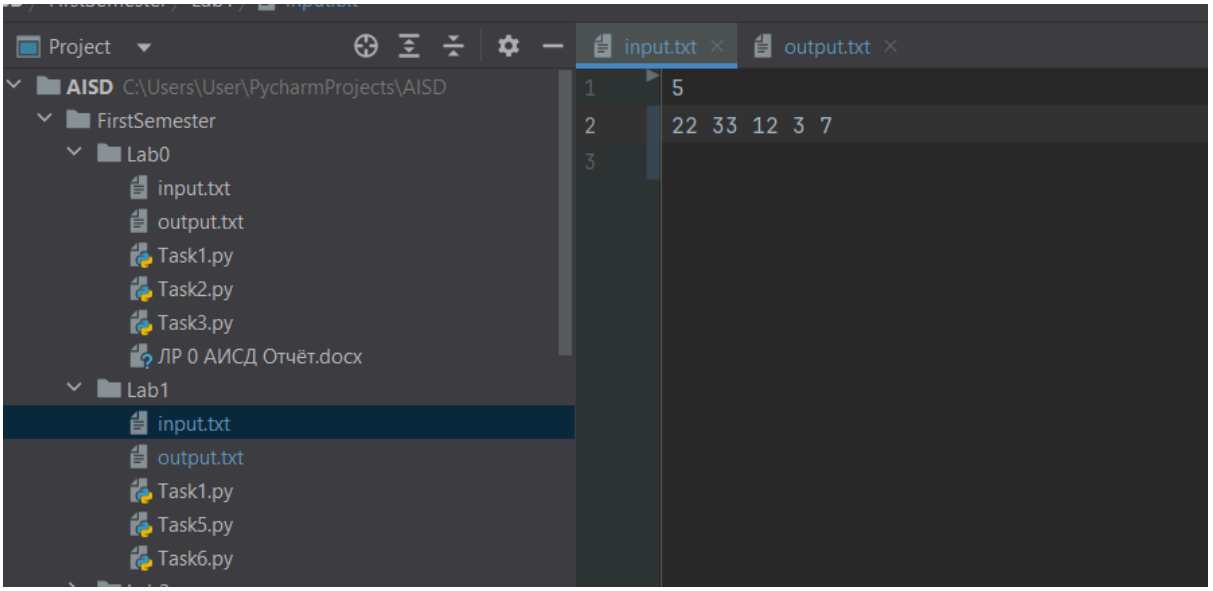
Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

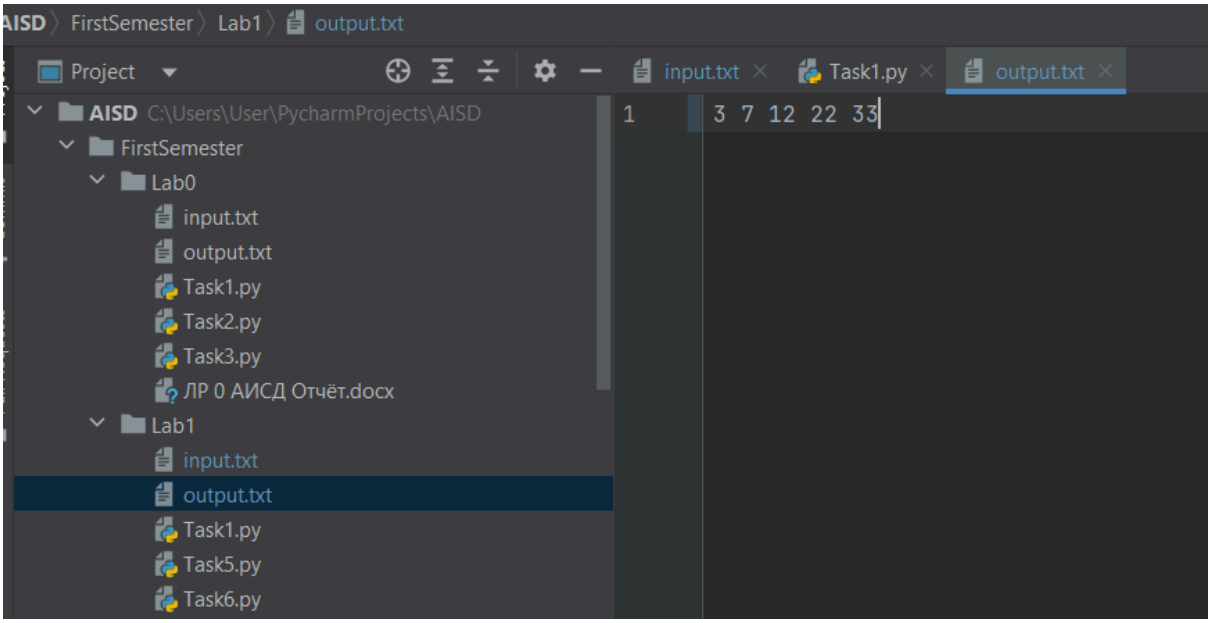
Код программы:

```
def insertion_sort(arr):
    # Реализация сортировки вставкой
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
         # Перемещение элементов, которые больше ключа,
на одну позицию вперед
```

```
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

def main():
    with open('input.txt', 'r') as f:
        n = int(f.readline().strip())
                                arr    =    list(map(int,
f.readline().strip().split()))

    # Сортировка массива
    insertion_sort(arr)

    with open('output.txt', 'w') as f:
        f.write(' '.join(map(str, arr)))

if __name__ == '__main__':
    main()
```

Результат работы кода на примерах:


Project
input.txt
output.txt
AISD C:\Users\User\PycharmProjects\AISD
FirstSemester
Lab0
input.txt
output.txt
Task1.py
Task2.py
Task3.py
ЛР 0 АИСД Отчёт.docx
Lab1
input.txt
output.txt
Task1.py
Task5.py
Task6.py
1
2
3
5
22 33 12 3 7


AISD
FirstSemester
Lab1
output.txt
Project
input.txt
Task1.py
output.txt
AISD C:\Users\User\PycharmProjects\AISD
FirstSemester
Lab0
input.txt
output.txt
Task1.py
Task2.py
Task3.py
ЛР 0 АИСД Отчёт.docx
Lab1
input.txt
output.txt
Task1.py
Task5.py
Task6.py
1
3 7 12 22 33

5 задача. Сортировка выбором.

Рассмотрим сортировку элементов массива , которая выполняется следующим образом. Сначала определяется наименьший элемент массива , который ставится на место элемента $A[1]$. Затем производится поиск второго наименьшего элемента массива A, который ставится на место элемента $A[2]$. Этот процесс продолжается для первых $n-1$ элементов массива A.

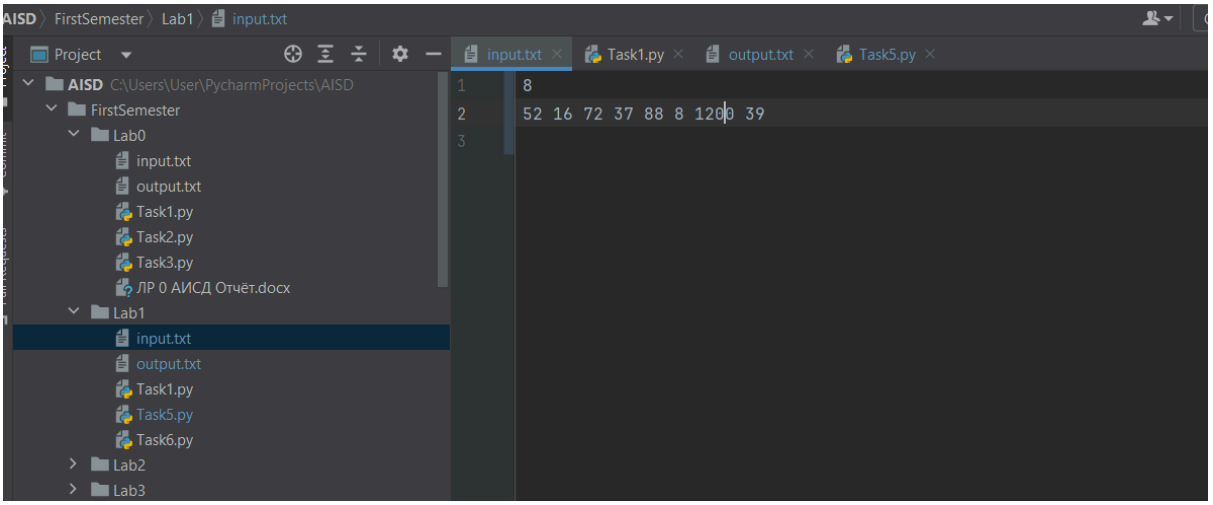
Напишите код этого алгоритма, также известного как сортировка выбором (selection sort). Определите время сортировки выбором в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

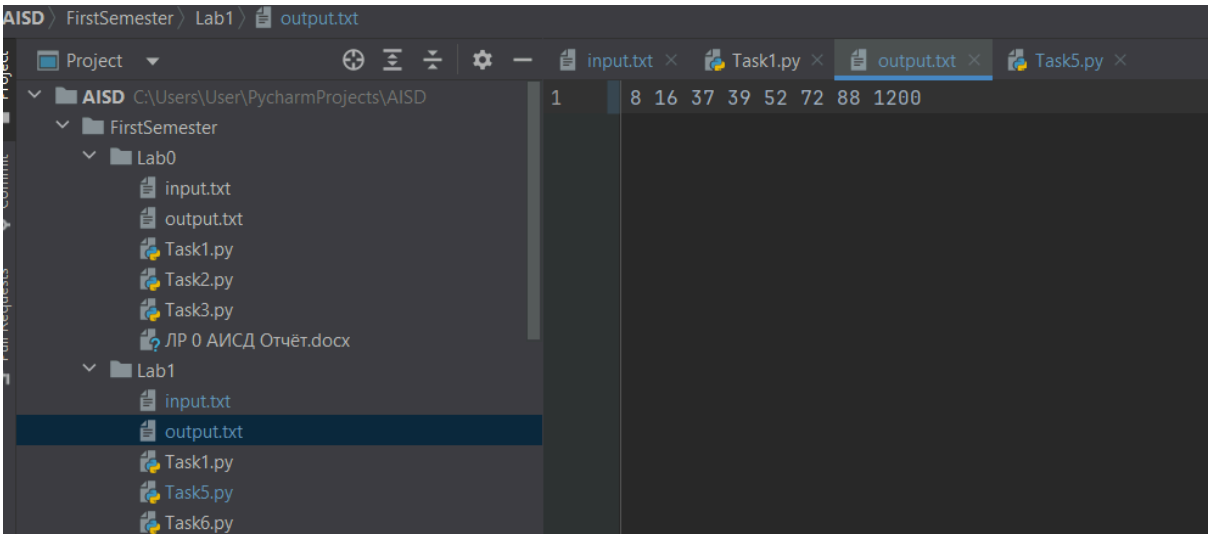
Формат входного и выходного файла и ограничения - как в задаче 1.

Код программы:

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

with open('input.txt') as f:
    arr = [int(line.strip()) for line in f]

# Сортировка
sorted_arr = selection_sort(arr)

with open('output.txt', 'w') as f:
    for num in sorted_arr:
        f.write(f"{num}\n")
```

Результат работы кода на примерах:


AISD FirstSemester Lab1 input.txt
Project
input.txt
Task1.py
output.txt
Task5.py
AISD C:\Users\User\PycharmProjects\AISD
FirstSemester
Lab0
input.txt
output.txt
Task1.py
Task2.py
Task3.py
ЛР 0 АИСД Отчёт.docx
Lab1
input.txt
output.txt
Task1.py
Task5.py
Task6.py
Lab2
Lab3
8
52 16 72 37 88 8 1200 39


AISD FirstSemester Lab1 output.txt
Project
input.txt
Task1.py
output.txt
Task5.py
AISD C:\Users\User\PycharmProjects\AISD
FirstSemester
Lab0
input.txt
output.txt
Task1.py
Task2.py
Task3.py
ЛР 0 АИСД Отчёт.docx
Lab1
input.txt
output.txt
Task1.py
Task5.py
Task6.py
8 16 37 39 52 72 88 1200

6 задача. Пузырьковая сортировка

Пузырьковая сортировка представляет собой популярный, но не очень эффективный алгоритм сортировки. В его основе лежит многократная перестановка соседних элементов, нарушающих порядок сортировки. Вот псевдокод этой сортировки:

```
Bubble_Sort(A):
  for i = 1 to A.length - 1
    for j = A.length downto i+1
      if A[j] < A[j-1]
        поменять A[j] и A[j-1] местами
```

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что $A'[1] \leq A'[2] \leq ... \leq A'[n]$, где A' - выход процедуры Bubble_Sort, а n - длина массива A.

Определите время пузырьковой сортировки в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

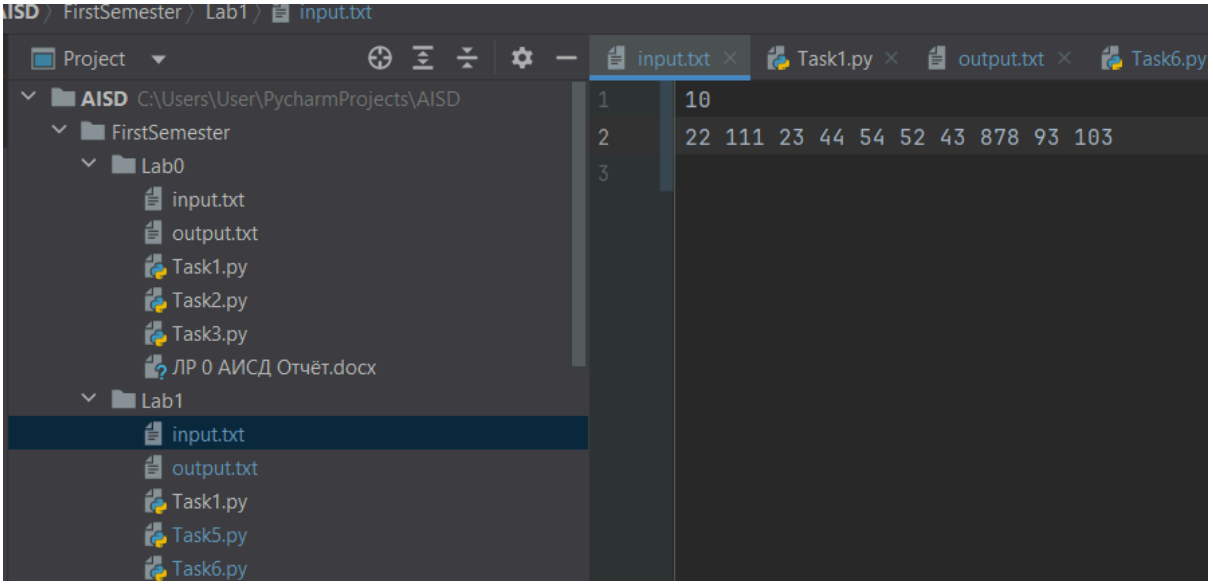
Формат входного и выходного файла и ограничения - как в задаче 1.

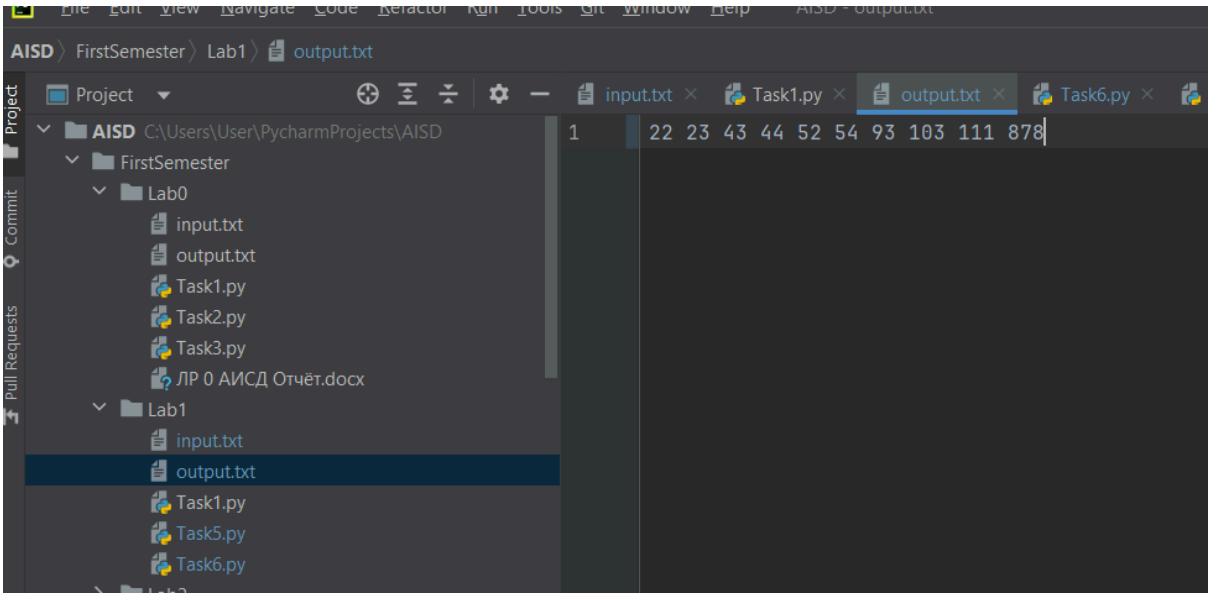
Код программы

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - 1, i, -1):
            if arr[j] < arr[j - 1]:
                    arr[j], arr[j - 1] = arr[j - 1],
arr[j]


def read_input(filename):
    with open(filename, "r") as file:
        n = int(file.readline().strip())
                                arr    =    list(map(int,
file.readline().strip().split()))
    return arr
```

```
def write_output(filename, arr):
    with open(filename, "w") as file:
        file.write(" ".join(map(str, arr)))

input_file = "input.txt"
output_file = "output.txt"
array = read_input(input_file)

# Выполнение сортировки
bubble_sort(array)

# Запись результата в выходной файл
write_output(output_file, array)
```

Результат работы кода на примерах:




AISD
FirstSemester
Lab1
output.txt
Project
input.txt
Task1.py
output.txt
Task6.py
AISD C:\Users\User\PycharmProjects\AISD
FirstSemester
Lab0
input.txt
output.txt
Task1.py
Task2.py
Task3.py
ЛР 0 АИСД Отчёт.docx
Lab1
input.txt
output.txt
Task1.py
Task5.py
Task6.py
1
22 23 43 44 52 54 93 103 111 878
Commit
Pull Requests