САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Алгоритмы и структуры данных»

Вариант 1

Выполнил:

Азаренков Георгий Денисович

К34421

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc179012155)

[Комментарий к общему коду 3](#_Toc179012156)

[Задача 1 4](#_Toc179012157)

[Текст задачи 4](#_Toc179012158)

[Код решения 4](#_Toc179012159)

[Результат выполнения 5](#_Toc179012160)

[Вывод 6](#_Toc179012161)

# Комментарий к общему коду

При использовании всех задач я использую код из файла “common.py”, который находится в корне репозитория.

from typing import List, Callable  
  
  
def solve(solver: Callable[[List[str]], List[str]]) -> None:  
 with open("../../input.txt", "r") as file:  
 lines = file.readlines()  
 input\_lines = list(map(lambda x: x.strip(), lines))  
  
 output\_lines = solver(input\_lines)  
  
 with open("../../output.txt", "w") as file:  
 for line in output\_lines:  
 file.write(f"{line}\n")

Использование функции “solve” позволило мне избавиться от необходимости раз за разом имплементировать логику чтения и сохранения данных.

# Задача 1

## Текст задачи

A white paper with black text

Description automatically generated

## Код решения

from common import solve  
  
  
def get\_solution(lines: list[str]) -> list[str]:  
 *"""  
 Программа сортирует массив целых чисел методом вставки и выводит отсортированный массив.  
 """* n = int(lines[0])  
  
 array = list(map(int, lines[1].split()))  
  
 *# Алгоритм сортировки вставками* for i in range(1, n):  
 *# Элемент, который будем вставлять* key = array[i]  
 j = i - 1  
  
 *# Сдвигаем элементы, которые больше ключа, вправо* while j >= 0 and array[j] > key:  
 array[j + 1] = array[j]  
 j -= 1  
  
 *# Вставляем ключ на нужное место* array[j + 1] = key  
  
 *# Преобразуем отсортированный массив обратно в строку с числами через пробел* sorted\_array\_str = " ".join(map(str, array))  
  
 *# Возвращаем результат в виде списка строк (одна строка)* return [sorted\_array\_str]  
  
  
solve(get\_solution)

## Результат выполнения

input.txt

A number on a white background

Description automatically generated

output.txt

A screenshot of a computer

Description automatically generated

# Задача 2

## Текст задачи

A screenshot of a computer

Description automatically generated

## Код решения

from typing import List  
  
from common import solve  
  
  
def get\_solution(input\_lines: List[str]) -> List[str]:  
 *# Читаем количество элементов n из первой строки* n = int(input\_lines[0])  
  
 *# Преобразуем вторую строку в список целых чисел* array = list(map(int, input\_lines[1].split()))  
  
 *# Инициализируем массив для хранения новых позиций элементов* positions = [0] \* n  
  
 *# Начинаем сортировку вставками* for i in range(n):  
 key = array[i] *# Текущий элемент для вставки* j = i - 1  
  
 *# Сохраняем текущий индекс элемента* original\_index = i  
  
 *# Перемещаем элементы, которые больше ключа, на одну позицию вперед* while j >= 0 and array[j] > key:  
 array[j + 1] = array[j] *# Сдвигаем элемент вправо* j -= 1  
  
 array[j + 1] = key *# Вставляем ключ на правильное место  
 # Новая позиция элемента после вставки* new\_position = j + 1 *# Индексы начинаются с 0* positions[original\_index] = new\_position + 1 *# Сохраняем новую позицию (с учетом индексации с 1)  
  
 # Преобразуем список позиций и отсортированный массив в строки с пробелами* positions\_line = " ".join(map(str, positions))  
 sorted\_array\_line = " ".join(map(str, array))  
  
 *# Возвращаем список строк для записи в output.txt* return [positions\_line, sorted\_array\_line]  
  
  
solve(get\_solution)

## Результаты выполнения

input.txt

A screenshot of a phone

Description automatically generated

output.txt

A number on a white background

Description automatically generated

A close up of numbers

Description automatically generated

# Задача 4

## Текст задачи

A black text on a white background

Description automatically generated

## Код решения

from typing import List  
  
from common import solve  
  
  
def get\_solution(input\_lines: List[str]) -> List[str]:  
 *"""  
 Функция выполняет линейный поиск значения V в массиве чисел  
 """  
  
 # Парсим первую строку в массив чисел* numbers = list(map(int, input\_lines[0].split()))  
  
 *# Парсим значение V из второй строки* V = int(input\_lines[1].strip())  
  
 *# Инициализируем список индексов, где найдено значение V* indices = []  
  
 *# Проходим по массиву чисел и ищем V* for i, num in enumerate(numbers):  
 if num == V:  
 indices.append(i + 1) *# Индексы начинаются с 1  
  
 # Если значение V не найдено* if not indices:  
 return ["-1"]  
  
 *# Если значение V найдено один раз* if len(indices) == 1:  
 return [str(indices[0])]  
  
 *# Если значение V найдено несколько раз* result = f"Number occurs {len(indices)} times at indices: {','.join(map(str, indices))}"  
  
 return [str(indices[0]), " ".join(map(str, indices))]  
  
  
solve(get\_solution)

## Результаты выполнения

input.txt

A number on a white background

Description automatically generated

output.txt

A close up of numbers

Description automatically generated

# Вывод

В ходе данной лабораторной работы я научился решать задачи по теме. Написанные программы были протестированы с измерением затрат времени и оперативной памяти. Все программы работаю корректно и укладываются в установленные ограничения.