Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Теория алгоритмов»

Тема: «Введение в разработку консольных приложений на python»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-21.06  Талакуцкая А.С. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2023

1. **Формулировка задачи, описание условий**

Для выполнения лабораторной работы необходимо:

1. Установить язык программирования и среду разработки.

2. Создать аккаунт и репозиторий на сайте https://github.com/

3. Выбрать 1 алгоритм сортировки и реализовать его на языке python

4. Изменения в программе коммитить в GIT по мере необходимости. После завершения разработки отправить изменения на сервер. В отчет приложить ссылку

В качестве реализуемого алгоритма выбрана сортировка вставками.

Технические требования:

1. Реализация на Python 3.\* (рекомендуемая версия python 3.8\*, учесть поддержку win7 и win8).

2. Вход осуществляется из файла, вывод осуществляется в файл.

3. Функции сортировки вынесена в def.

4. Сравнить время выполнения алгоритмов на тестовых выборках разного размера. Построить графики.

1. **Алгоритм решения лабораторной работы**

Сортировка вставками (англ. Insertion sort) — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. Вычислительная сложность — O(n2).

**Описание алгоритма**

На вход алгоритма подаётся последовательность n чисел: a1, a2, ..., an. Сортируемые числа также называют ключами. Входная последовательность на практике представляется в виде массива с n элементами. На выходе алгоритм должен вернуть перестановку исходной последовательности a’1, a’2, …, a’n, чтобы выполнялось следующее соотношение a’1<= a’2<= … <= a’n.

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

**Псевдокод**

На вход процедуре сортировки подаётся массив A[1...n] состоящий из элементов последовательности A[1],A[2],...,A[n], которые требуется отсортировать. n соответствует A.length — размеру исходного массива. Для сортировки не требуется привлечения дополнительной памяти, кроме постоянной величины для одного элемента, так как выполняется перестановка в пределах массива. В результате работы процедуры во входном массиве оказывается требуемая выходная последовательность элементов.

for j = 2 to A.length do

key = A[j]

i = j-1

while (i >= 0 and A[i] > key) do

A[i + 1] = A[i]

i = i - 1

end while

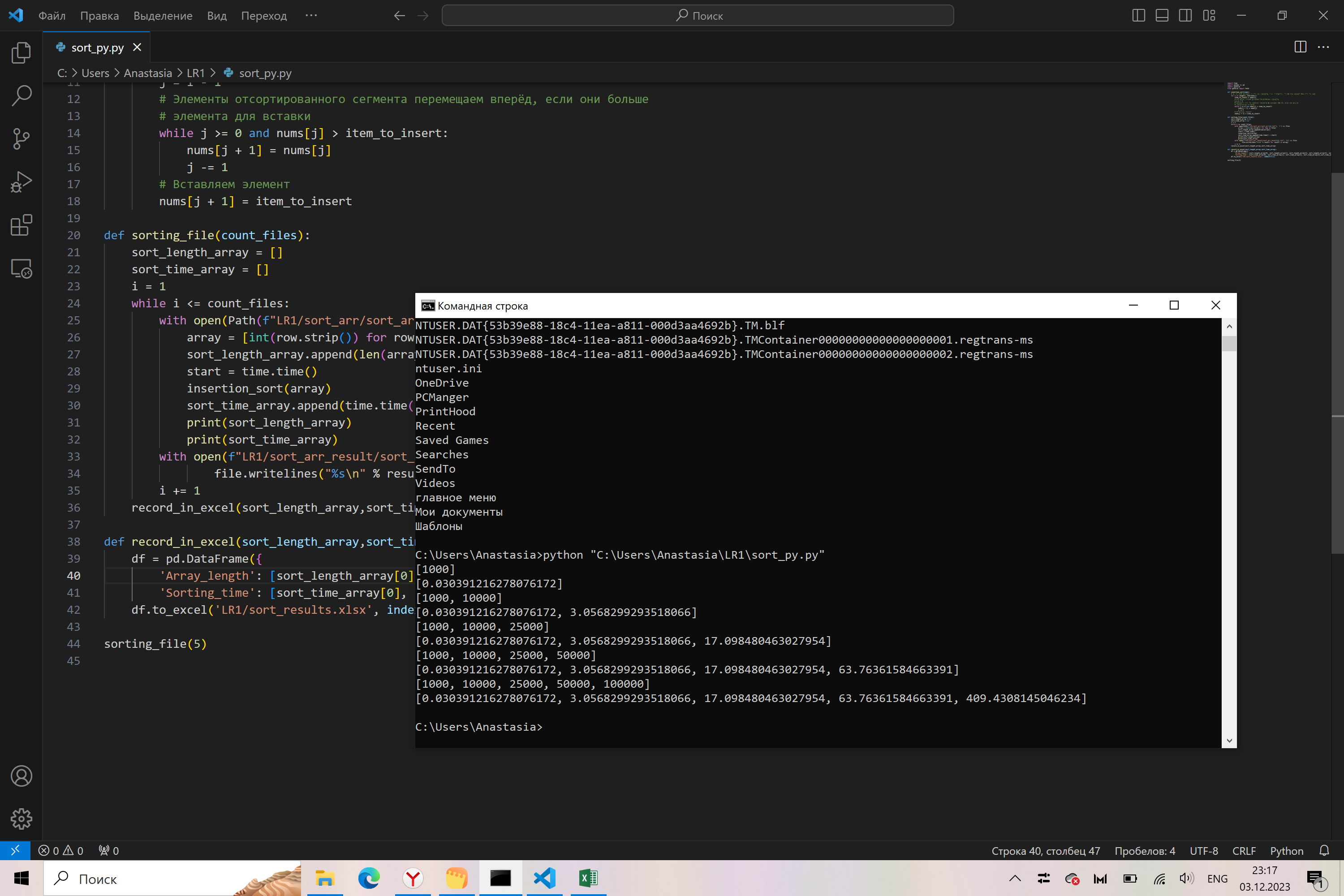
A[i+1] = key

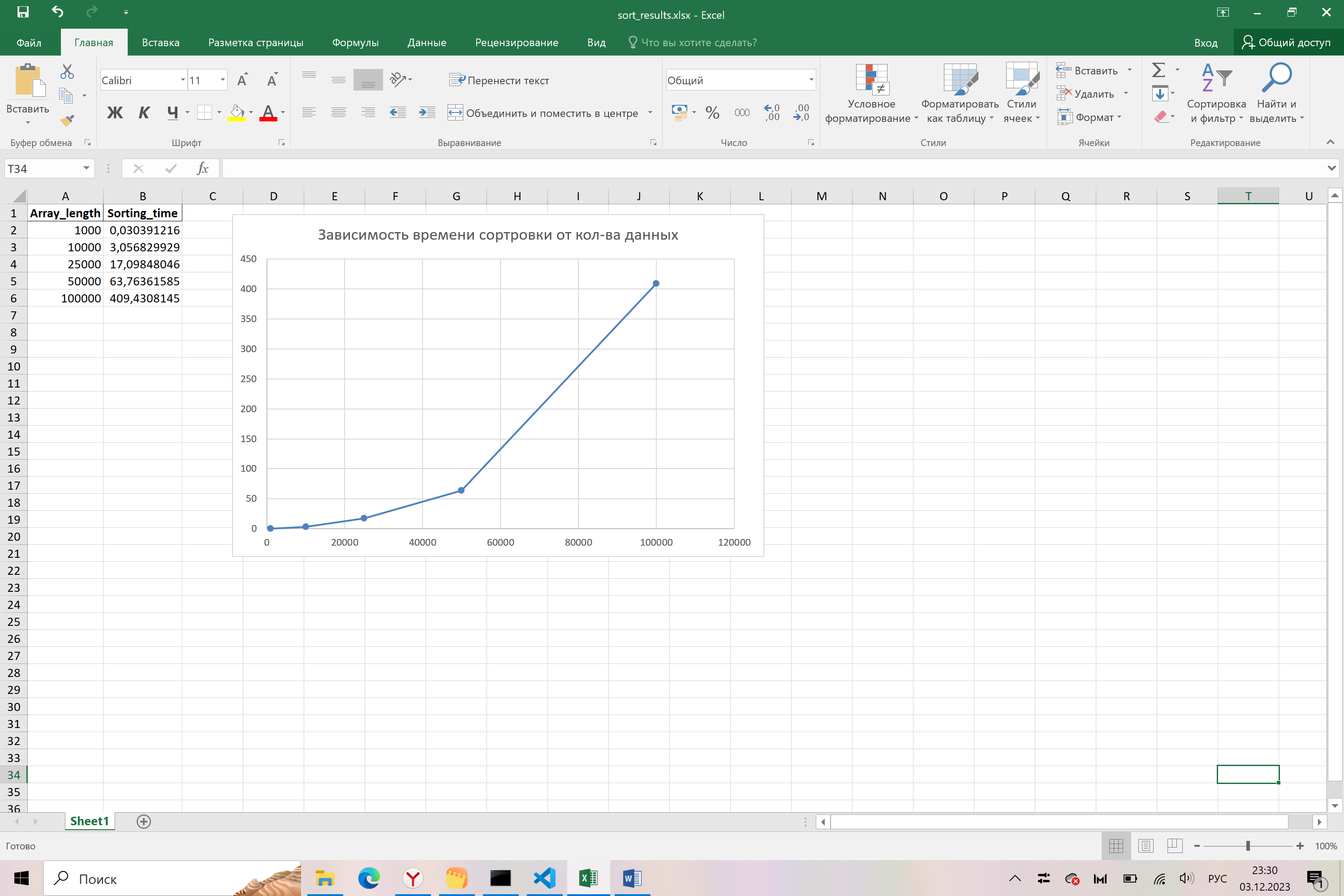
end

1. **Тестирование программы**
   1. **Описание тестового случая**

Для тестирования программы было создано пять файлов, содержащих неупорядоченные массивы, размерами 1000, 10000, 25000, 50000 и 100 000 и с диапазоном значений от 0 до 100. В ходе работы программы отсортированные массивы записываются в соответствующие файлы. Результат времени работы заносится в таблицу Excel.

* 1. **Скриншоты**





1. **Пример кода**

import time

import pandas as pd

import pathlib

from pathlib import Path

def insertion\_sort(nums):

# Сортировку начинаем со второго элемента, т.к. считается, что первый элемент уже отсортирован

for i in range(1, len(nums)):

item\_to\_insert = nums[i]

# Сохраняем ссылку на индекс предыдущего элемента

j = i - 1

# Элементы отсортированного сегмента перемещаем вперёд, если они больше

# элемента для вставки

while j >= 0 and nums[j] > item\_to\_insert:

nums[j + 1] = nums[j]

j -= 1

# Вставляем элемент

nums[j + 1] = item\_to\_insert

def sorting\_file(count\_files):

sort\_length\_array = []

sort\_time\_array = []

i = 1

while i <= count\_files:

with open(Path(f"LR1/sort\_arr/sort\_arr{i}.txt"), 'r') as file:

array = [int(row.strip()) for row in file]

sort\_length\_array.append(len(array))

start = time.time()

insertion\_sort(array)

sort\_time\_array.append(time.time() - start)

print(sort\_length\_array)

print(sort\_time\_array)

with open(f"LR1/sort\_arr\_result/sort\_arr\_result{i}.txt", "w") as file:

file.writelines("%s\n" % result for result in array)

i += 1

record\_in\_excel(sort\_length\_array,sort\_time\_array)

def record\_in\_excel(sort\_length\_array,sort\_time\_array):

df = pd.DataFrame({

'Array\_length': [sort\_length\_array[0], sort\_length\_array[1], sort\_length\_array[2], sort\_length\_array[3], sort\_length\_array[4]],

'Sorting\_time': [sort\_time\_array[0], sort\_time\_array[1], sort\_time\_array[2], sort\_time\_array[3],sort\_time\_array[4]]})

df.to\_excel('LR1/sort\_results.xlsx', index=False)

sorting\_file(5)

1. **Вывод: описание реализованной программы и её функций**

В ходе выполнения лабораторной работы по реализации сортировки вставками были написаны следующие функции:

* insertion\_sort(nums) – функция, реализующая алгоритм сортировки вставками. На вход принимает неупорядоченный массив данных.
* sorting\_file(count\_files) – функция, осуществляющая чтение неупорядоченных массивов из документов, вызов функции сортировки (insertion\_sort), подсчёт времени сортировки и сохранение его в массив. Подсчёт времени начинается перед вызовом функции сортировки, а заканчивается после её выполнения. На вход принимает количество файлов с неотсортированными массивами.
* record\_in\_excel(sort\_length\_array, sort\_time\_array) – функция, для генерирования отчёта по работе функции сортировки массива алгоритмом сортировки вставками в формате таблицы Excel при помощи библиотеки pandas. На вход принимает два массива: sort\_length\_array – длины отсортированных массивов, sort\_time\_array – соответствующее каждой длине массива время сортировки.

Для запуска программы необходимо иметь файлы с неупорядоченными массивами, а также передать на вход количество этих файлов.

Программа работает корректно.