МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника» " " 20 г. Заведующий кафедрой

М.А. Митрохин

**ОТЧЕТ ПО УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ**

(2024/2025 учебный год)

Кузьмичев Александр Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника» " " 20 г. Заведующий кафедрой

М.А. Митрохин

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2024/2025 учебный год)

Кузьмичев Александр Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года Год обучения 1 семестр 2 Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая  форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения  работ | 2 | 25.06.2025 -  25.06.2025 |  |
| 2 | Подбор и изучение  материала по теме работы | 15 | 26.06.2025 –  27.06.25 |  |
| 3 | Разработка  алгоритма | 43 | 27.06.25 –  02.07.25 |  |
| 4 | Описание  алгоритма и программы | 18 | 02.07.25 –  04.07.25 |  |
| 5 | Тестирование | 5 | 05.07.25 –  05.07.25 |  |
| 6 | Получение и  анализ результатов | 10 | 05.07.25 –  06.07.25 |  |
| 7 | Оформление  отчёта | 15 | 06.07.25 –  08.07.2025 |  |
|  | **Общий объём**  **часов** | 108 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2024/2025 учебный год)

Кузьмичев Александр Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

Кузьмичев А.А. выполнял практическое задание «Сортировка вставками».

На первоначальном этапе изучил и проанализировал алгоритм сортировки вставками, после чего был выбран метод решения и язык программирования Python. Реализовал собственную версию алгоритма шейкерной сортировки, а также графический интерфейс программы с возможностью загрузки и сохранения данных из файлов.

Осуществил работу с CSV-файлами, реализовал визуализацию шагов сортировки, подключение GUI через библиотеку Tkinter, а также вывод времени выполнения и количества операций. Провёл тестирование и отладку программы, устранил найденные ошибки, оформил итоговый отчёт.

Бакалавр Тумасов В.В. " " 2025

г.

Руководитель Зинкин С.А. " " 2025 г. практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2024/2025 учебный год)

Кузьмичев Алексадр Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

В процессе выполнения практики Кузьмичев А.А. решал следующие задачи: создание алгоритма шейкерной сортировки, анализ работы алгоритма.

За период выполнения практики были освоены основные понятия и технологии шейкерной сортировки, реализованы метод работы с файлами и сохранение файлов. Во время выполнения работы Кузьмичев А.А. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Кузьмичев А.А. заслуживает оценки « ».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2025 г.

Содержание

[Введение 2](#_Toc1)

[1 Постановка задачи 4](#_Toc2)

[1.1 Достоинства алгоритма шейкерной сортировки 4](#_Toc3)

[1.2 Недостатки алгоритма шейкерной сортировки 4](#_Toc4)

[1.3 Типичные сценарии применения данного алгоритма 4](#_Toc5)

[2 Выбор решения 6](#_Toc6)

[3 Описание программы 7](#_Toc7)

[4. Схемы программы 9](#_Toc8)

[4.1 Блок-схема программы 9](#_Toc9)

[4.2 Блок-схема алгоритма 10](#_Toc10)

[5 Тестирование программы 11](#_Toc11)

[6 Отладка 12](#_Toc12)

[Заключение 15](#_Toc13)

[Список используемой литературы 16](#_Toc14)

[Приложение А. Листинг программы 17](#_Toc15)

# Введение

Сортировка данных — один из наиболее важных и часто встречающихся процессов в современных информационных технологиях. Практически в каждой профессиональной сфере обработка данных включает в себя этап их сортировки, будь то базы данных, обработка пользовательских запросов, статистика или визуализация. Без предварительной сортировки многие операции, в частности поиск, выполнялись бы медленнее и неэффективно.

Алгоритмы сортировки представляют собой отдельный класс алгоритмов, используемых для упорядочивания элементов по определённому критерию. Они служат не только практическим задачам, но и широко применяются в образовательных целях, так как позволяют наглядно изучить фундаментальные принципы построения алгоритмов.

Шейкерная сортировка (или сортировка перемешиванием, двунаправленная пузырьковая сортировка) является усовершенствованием классического пузырькового метода. В отличие от последнего, шейкерная сортировка проходит по массиву в обоих направлениях, что позволяет быстрее перемещать как минимальные, так и максимальные элементы к границам массива. Это улучшение снижает количество проходов и увеличивает эффективность алгоритма на частично отсортированных массивах.

Несмотря на то что шейкерная сортировка не является самой производительной по сравнению с более сложными алгоритмами, такими как быстрая сортировка или Timsort, она остаётся полезной для изучения и наглядного понимания принципов сортировки. Кроме того, она применима в тех случаях, когда важна простота реализации и визуализация работы алгоритма.

Шейкерная сортировка демонстрирует важную идею оптимизации уже существующего алгоритма за счёт незначительных изменений в логике его выполнения. Это делает её удобным примером для анализа, тестирования и сравнения с другими методами сортировки.

# Постановка задачи

Поставленная задача: необходимо реализовать алгоритм шейкерной сортировки, позволяющий упорядочить одномерный массив из n элементов. Массив может быть введён пользователем вручную или считан из CSV-файла. После сортировки данные должны быть сохранены в отдельный файл. Программа должна отображать отсортированный массив, количество перестановок, время выполнения алгоритма и визуализировать шаги сортировки в графическом интерфейсе.

Использовать сервис GitHub для совместной работы. Создать и выложить коммиты, характеризующие действия, выполненные каждым участником бригады.

Оформить отчет по проведенной практике.

## Достоинства алгоритма шейкерной сортировки

* Алгоритм лучше стандартной пузырьковой сортировки благодаря проходу в двух направлениях;
* Хорошо работает на почти отсортированных данных;
* Прост в реализации и визуализации;
* Позволяет быстрее перемещать крупные и мелкие элементы к своим позициям.

## Недостатки алгоритма шейкерной сортировки

* Неэффективен на больших массивах;
* Имеет высокую алгоритмическую сложность O(n²);
* Много лишних проходов и сравнений при работе с неупорядоченными данными;
* Не используется в промышленном программировании — подходит в основном для обучения.

## Типичные сценарии применения данного алгоритма

* Учебные проекты по анализу и визуализации алгоритмов;
* Работа с небольшими массивами, например:
* список заказов (по дате, стоимости);
* студенты в группе (по среднему баллу, фамилии);
* книги в библиотеке (по названию, дате выпуска);

# Выбор решения

Нашей бригадой было выбрано вести разработку в среде Microsoft Visual Studio с использованием языка программирования Python. Данная среда поддерживает Python и предоставляет удобный интерфейс для написания, отладки и запуска кода, включая работу с графическим интерфейсом и подключением внешних библиотек.

Язык программирования Python был выбран благодаря своей простоте, читаемости, высокой скорости разработки и встроенной поддержке работы с файлами, структурами данных и графическим интерфейсом. Python широко используется как в образовательных, так и в промышленных проектах, благодаря своей эффективности и универсальности.

Для реализации графического интерфейса программы была использована встроенная библиотека Tkinter, входящая в стандартную поставку Python. Она позволяет быстро создавать окна, поля ввода, кнопки и другие элементы взаимодействия с пользователем.

Для удобства совместной разработки и хранения версий проекта использовался сервис GitHub. Каждый участник команды вносил свой вклад, фиксируя изменения в виде коммитов, что позволяло отслеживать прогресс, быстро выявлять ошибки и координировать действия участников.

# Описание программы

При запуске программы открывается графическое окно, содержащее следующие элементы:

* поле ввода для ввода массива вручную (через запятую);
* кнопка загрузки массива из CSV-файла;
* кнопка запуска сортировки шейкером;
* кнопка сохранения отсортированного массива в файл;
* текстовое поле для отображения результата сортировки и времени её выполнения;
* отдельный блок, где пошагово отображаются действия алгоритма сортировки.

После ввода данных (или загрузки из CSV), программа:

* Проверяет корректность введённого массива;
* Запускает алгоритм шейкерной сортировки;
* Выводит отсортированный массив и время сортировки с точностью до миллионных долей секунды;
* Показывает все шаги работы алгоритма, включая направление прохода и сравниваемые значения;
* Позволяет сохранить результат в файл.

В рамках сортировки массив перебирается поочерёдно в двух направлениях: слева направо и справа налево. За один полный проход в обе стороны происходит постепенное «выталкивание» минимальных и максимальных элементов к своим позициям. Это ускоряет сортировку по сравнению с обычной пузырьковой.

Дополнительно программа считает общее количество перестановок элементов, а также измеряет время выполнения сортировки.

Если пользователь не вводит данные вручную, можно загрузить массив из CSV-файла. Также реализована проверка на наличие ошибок (например, буквы вместо чисел), и в случае некорректного ввода появляется соответствующее сообщение.

При завершении работы пользователь может сохранить результат в файл, либо просто закрыть окно.

Подробный алгоритм сортировки представлен в подразделе 4.1, на рисунках 1 и 2.

Листинг программы приведён в Приложении А.

# Схемы программы

## Блок-схема программы

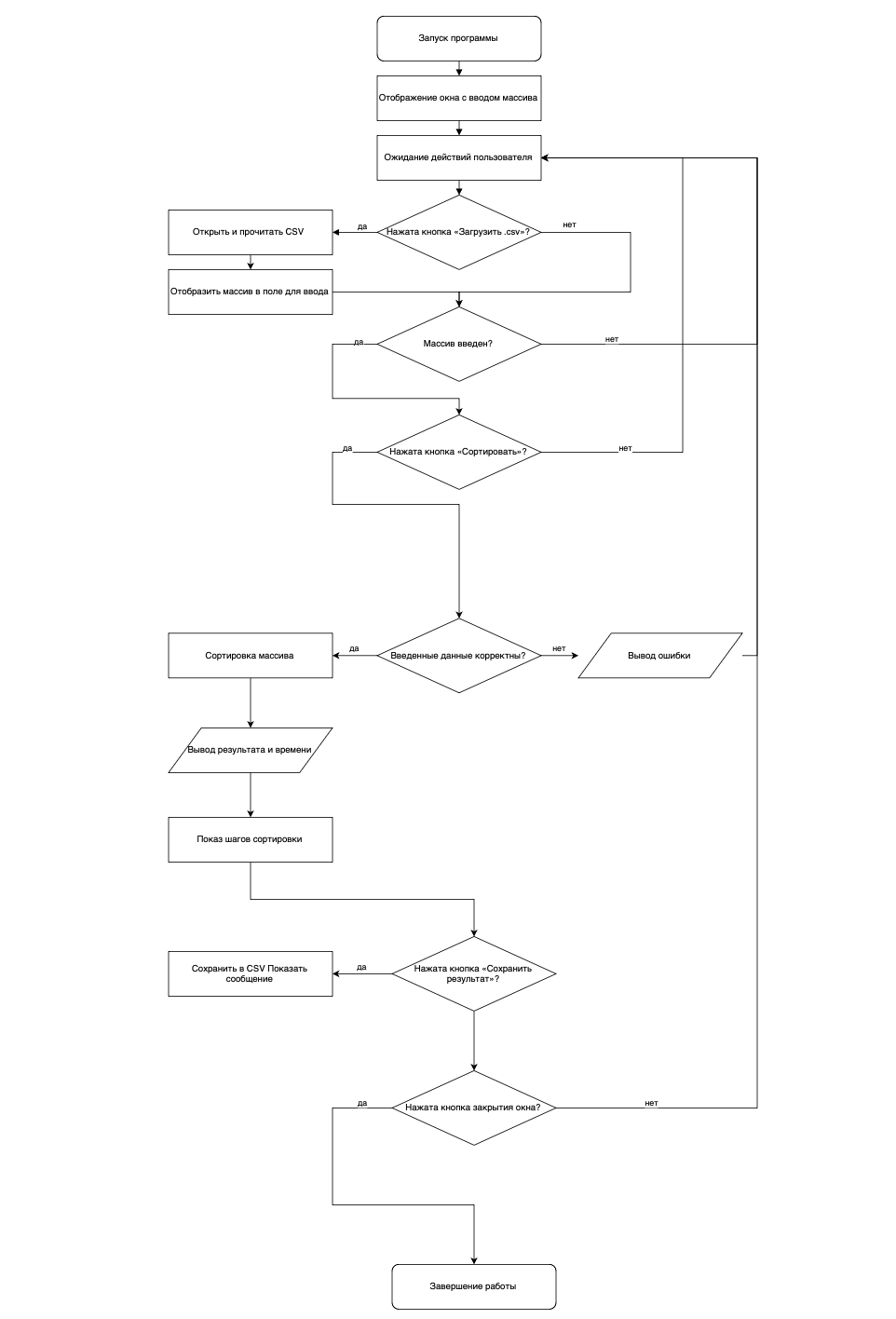


Рисунок 1 **-** Блок-схема программы

## Блок-схема алгоритма

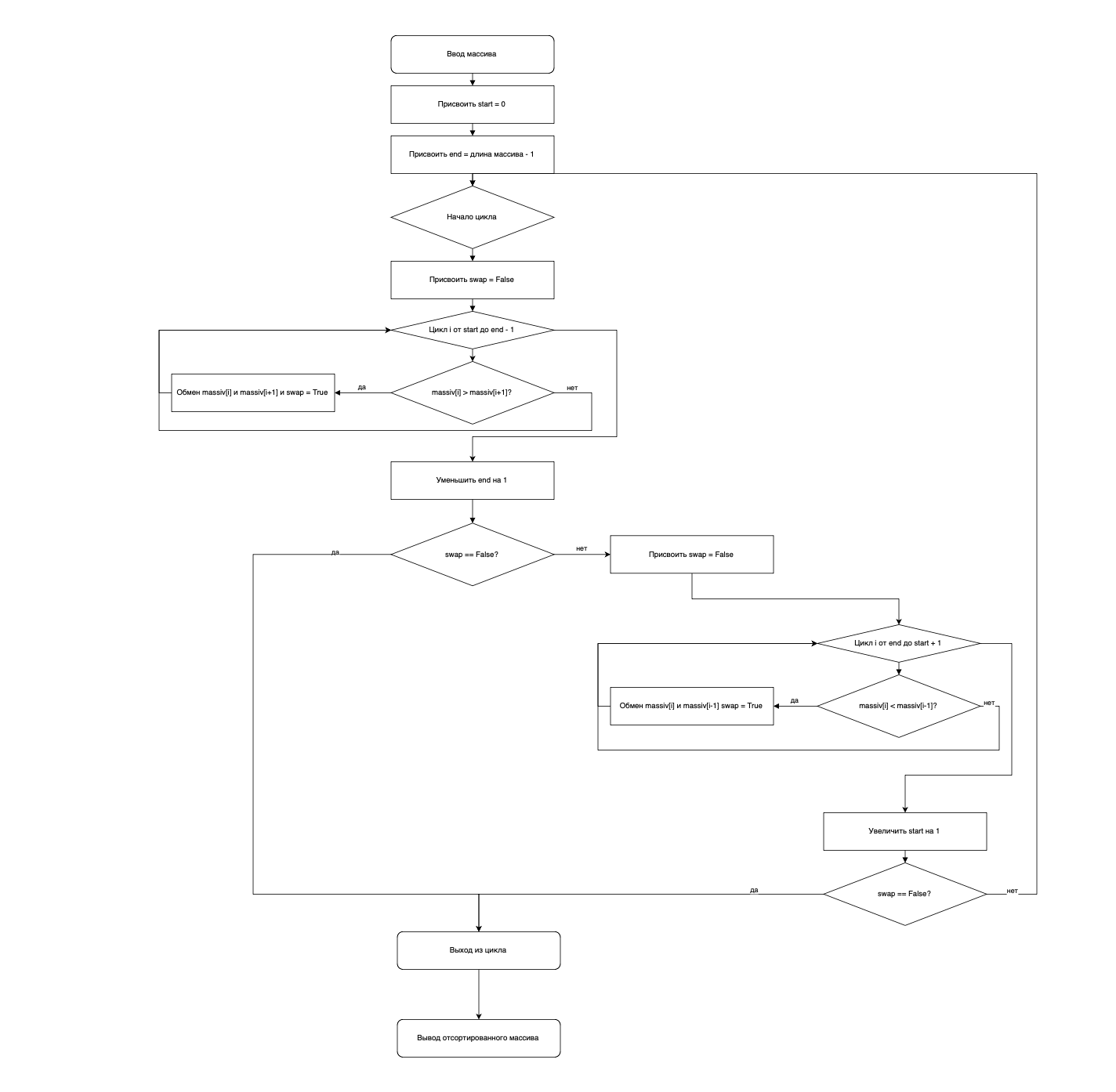


Рисунок 2 - Блок-схема алгоритма сортировки

# Тестирование программы

Тестирование показало, что с увеличением количества элементов пропорционально увеличивается время работы программы, ниже представлен график результатов тестирования

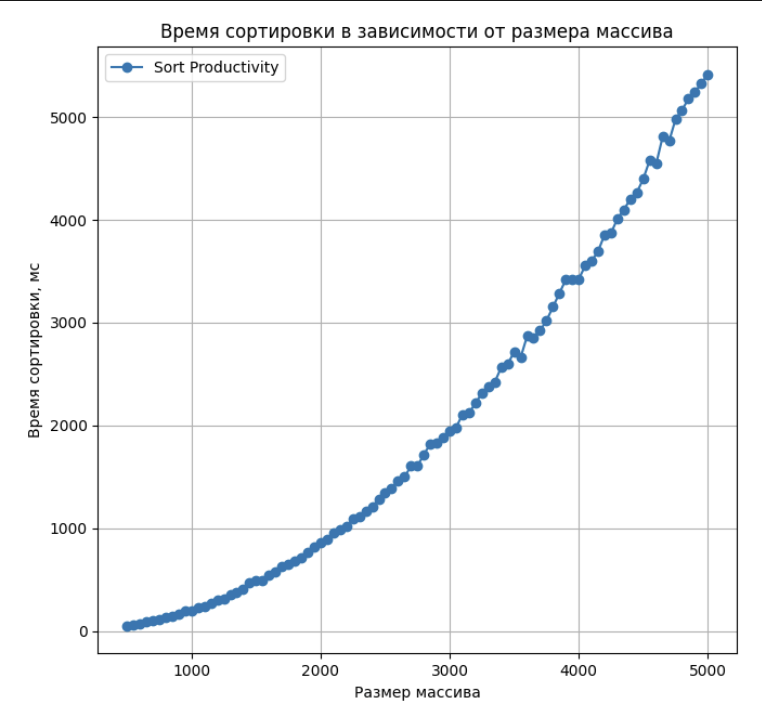


Рисунок 3 – Результаты тестирования

# Отладка

В качестве среды разработки использовалась Microsoft Visual Studio с установленным расширением Python, которое предоставляет все необходимые инструменты для написания, запуска и отладки Python-программ.

Для поиска и устранения ошибок применялись следующие методы отладки:

* Точки останова (breakpoints) — использовались для остановки выполнения программы в ключевых местах, что позволило пошагово отслеживать работу алгоритма сортировки и взаимодействие с пользовательским интерфейсом.
* Пошаговое выполнение кода (step over, step into) — применялось для поочерёдного прохождения инструкций программы, что позволило детально отслеживать, как изменяется состояние переменных и массива на каждом этапе.
* Просмотр значений локальных переменных — в процессе отладки изучалось содержимое массивов, входных строк и флагов (swap и др.) для контроля логики выполнения сортировки.

Отладка особенно активно применялась при разработке следующих частей:

* функции shaker\_sort(), чтобы проверить корректность перестановок элементов;
* функции shaker\_sort\_steps(), чтобы убедиться в правильности генерации текста пошаговых действий;
* обработки ошибок при загрузке файла и парсинге чисел;
* визуализации результата и интерфейса Tkinter.

Также применялся метод "деления и локализации" — изолированное тестирование отдельных функций и блоков кода. Например, перед подключением сортировки к интерфейсу, алгоритм отлаживался в консольной версии для исключения логических ошибок.

Команда «Step Into» (шаг с заходом) активно использовалась при отладке функций сортировки и обработки массива, чтобы переходить внутрь пользовательских функций (shaker\_sort, input\_check, upload\_csv) и отслеживать их выполнение строка за строкой.

# Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервисов GitHub и WEEK, навыки использования программы Git Bash. Был изучен алгоритм сортировки вставками.

Мною было создано меню программы, позволяющее выбрать одну из нескольких опций программы. Также я осуществила подключение файлов к программе, в которые выводятся результаты сгенерированного и отсортированного массивов.

При выполнении практической работы были улучшены базовые навыки программирования на языке С. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

В дальнейшем программу можно улучшить путем подключения упрощающих реализацию данной сортировки библиотек и улучшения графического интерфейса.

# Список используемой литературы

1. ГОСТ 19.701–90. Схемы алгоритмов, программ, данных и систем. – М.: Издательство стандартов, 1991.
2. Cocktail shaker sort [Электронный ресурс] – Wikipedia. – URL: https://ru.wikipedia.org/wiki/Сортировка\_перемешиванием (дата обращения: 07.07.2025 г.) ru.wikipedia.org+8ru.wikipedia.org+8wikidata.org+8
3. Изучение различных видов сортировок [Электронный ресурс] – Habr. – URL: https://habr.com/ru/articles/850302/ (дата обращения: 07.07.2025 г.)
4. tkinter — графические интерфейсы на Python [Электронный ресурс] – официальная документация Python. – URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 07.07.2025 г.)anzeljg.github.io+3docs.python.org+3docs.python.org+3
5. Python Tkinter Tutorial [Электронный ресурс] – GeeksforGeeks (обзор использования tkinter). – URL: https://www.geeksforgeeks.org/python/python-gui-tkinter/ (дата обращения: 07.07.2025 г.) geeksforgeeks.org

# Приложение А. Листинг программы

import tkinter as tk

from tkinter import filedialog, messagebox

import csv

import time

def shaker\_sort(massiv):

start = 0

end = len(massiv) - 1

while True:

swap = False

for i in range(start, end):

current\_element = massiv[i]

next\_element = massiv[i + 1]

if current\_element > next\_element:

massiv[i] = next\_element

massiv[i + 1] = current\_element

swap = True

end -= 1

if swap == False:

break

swap = False

for i in range(end, start, -1):

current\_element = massiv[i]

next\_element = massiv[i - 1]

if current\_element < next\_element:

massiv[i] = next\_element

massiv[i - 1] = current\_element

swap = True

start += 1

if swap == False:

break

return massiv

def shaker\_sort\_steps(massiv):

steps = []

start = 0

end = len(massiv) - 1

while True:

swap = False

steps.append("проход вперед:")

for i in range(start, end):

current\_element = massiv[i]

next\_element = massiv[i + 1]

if current\_element > next\_element:

massiv[i] = next\_element

massiv[i + 1] = current\_element

swap = True

steps.append(f"{massiv} – {current\_element} > {next\_element}?")

end -= 1

steps.append("проход назад:")

if swap == False:

break

swap = False

for i in range(end, start, -1):

current\_element = massiv[i]

next\_element = massiv[i - 1]

if current\_element < next\_element:

massiv[i] = next\_element

massiv[i - 1] = current\_element

swap = True

steps.append(f"{massiv} – {current\_element} < {next\_element}?")

start += 1

if swap == False:

break

return steps

def input\_check(input\_data):

input\_data = input\_data.split(",")

massiv = []

for i in input\_data:

try:

massiv.append(int(i.strip()))

except ValueError:

messagebox.showerror("Ошибка", "Введите числа через запятую")

return False

return massiv

def array\_sort():

global last\_sorted\_massiv

input\_str = input\_data.get()

massiv = input\_check(input\_str)

if massiv == False:

return False

else:

start\_time = time.time()

sorted\_massiv = shaker\_sort(massiv.copy()) #чтобы не изменять оригинал

total\_time = time.time() - start\_time

output.config(text=f"{sorted\_massiv} \n\n Время сортировки: {total\_time:.6f} секунд ")

last\_sorted\_massiv = sorted\_massiv # Сохраняем результат для сохранения в файл

return show\_steps(massiv)

def show\_steps(massiv):

steps\_list = shaker\_sort\_steps(massiv)

steps\_text = "\n\n".join(map(str, steps\_list)) #преобразуем список содержащий строки и списки с числами в строку

steps.config(text=f"{steps\_text}")

def upload\_csv():

file\_path = filedialog.askopenfilename(filetypes=[("CSV Files", "\*.csv")])

if not file\_path:

if not file\_path:

return False

try:

with open(file\_path, "r") as file:

reader = csv.reader(file)

values = []

for row in reader:

for item in row:

values.append(int(item.strip()))

input\_data.delete(0, tk.END)

input\_data.insert(0, ",".join(map(str, values)))

except Exception as e:

messagebox.showerror("Ошибка при загрузке CSV", str(e))

def save\_csv():

if not last\_sorted\_massiv:

messagebox.showwarning("Нет данных", "Сначала отсортируйте массив.")

return False

file\_path = filedialog.asksaveasfilename(defaultextension=".csv",

filetypes=[("CSV Files", "\*.csv")],

title="Сохранить отсортированный массив")

if file\_path:

try:

with open(file\_path, "w", newline="") as file:

writer = csv.writer(file)

writer.writerow(last\_sorted\_massiv)

messagebox.showinfo("Успешно", "Результат сохранён.")

except Exception as e:

messagebox.showerror("Ошибка при сохранении", str(e))

window = tk.Tk()

window.title("Сортировка шейкером")

window.geometry("500x250")

last\_sorted\_massiv = []

tk.Label(window, text="Введите массив (через запятую): ").pack(pady=5)

input\_data = tk.Entry(window, width=60)

input\_data.pack(pady=5)

tk.Button(window, text="Загрузить .csv", command=upload\_csv).pack(pady=5)

tk.Button(window, text="Сортировать", command=array\_sort).pack(pady=5)

output\_label = tk.Label(window, text="Результат: ")

output\_label.pack(pady=10)

output = tk.Label(window, text="")

output.pack(pady=10)

steps\_label = tk.Label(window, text="Шаги сортировки:")

steps\_label.pack(pady=20)

steps = tk.Label(window, text="", justify="left")

steps.pack(pady=5)

tk.Button(window, text="Сохранить результат", command=save\_csv).pack(pady=5)

window.mainloop()