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)

Кукушкин Антон Алексеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.2025 по 8.07.2025

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)

                                          Кукушкин Антон Алексеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения                  1                 семестр                 2

Период прохождения практики с 25.06.2025 по 8.07.2025

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 –  28.06.25 |  |
| 3 | Разработка алгоритма | 43 | 28.06.25 –  02.07.25 |  |
| 4 | Описание алгоритма и программы | 18 | 02.07.25 –  04.07.25 |  |
| 5 | Тестирование | 5 | 04.07.25–  04.07.25 |  |
| 6 | Получение и анализ результатов | 10 | 04.07.25 –  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)

                                Кукушкиным Антоном Алексеевичем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.2025 по 8.07.2025

Кафедра «Вычислительная техника»

В рамках практического задания по теме "Сортировка пузырьком" Кукушкиным А.А. изначально был проведён детальный анализ принципов работы алгоритма пузырьковой сортировки. На основе полученных знаний был определён оптимальный подход к решению поставленной задачи и выбран язык программирования Python в качестве инструмента реализации.

В процессе работы была обеспечена функциональность взаимодействия с файлами. Также был разработан пользовательский интерфейс. Результаты выполненной работы были задокументированы в форме отчёта.

Бакалавр Кукушкин А.А. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                Кукушкин Антон Алексеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.2025 по 8.07.2025

Кафедра «Вычислительная техника»

В ходе прохождения практики Кукушкин А.А. занимался решением ряда задач: изучение принципов функционирования алгоритма сортировки пузырьком, работа с файловой системой, создание графического интерфейса пользователя.

За время прохождения практики были усвоены основные понятия, связанные с пузырьковой сортировкой, а также реализована методика взаимодействия с файлами. В процессе выполнения поставленных задач Кукушкин 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 Кукушкин А.А. заслуживает оценки «\_\_\_\_\_\_\_\_\_\_».

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

Оглавление

**[1. Постановка задачи 10](#_Toc201784480)**

**[1.1 Принцип работы 10](#_Toc201784482)**

**[1.2 Достоинства алгоритма 11](#_Toc201784483)**

**[1.3 Недостатки алгоритма 11](#_Toc201784484)**

**[1.4 Типичные сценарии использования 11](#_Toc201784485)**

**[2. Выбор решения 12](#_Toc201784486)**

**[3. Описание программы 13](#_Toc201784487)**

**[4. Схема алгоритма 19](#_Toc201784488)**

**[5. Отладка 21](#_Toc201784489)**

**[6. Совместная разработка 22](#_Toc201784490)**

**[Заключение 25](#_Toc201784491)**

**[Список использованных источников 26](#_Toc201784492)**

**[Приложение Б. Листинг программы. 30](#_Toc201784493)**

# Введение

В нынешнюю эпоху информационные технологии глубоко проникли в повседневную жизнь практически каждого человека. Сокращение трудозатрат в любой отрасли, ускорение общественных процессов и, как следствие, повышение их результативности в настоящее время обусловлены значительно расширившимися за последние десятилетия возможностями по сбору, обработке, хранению, передаче и представлению данных.

Использование информации качественно нового уровня, полученной в результате этих процессов, позволяет поднять создание и применение материальных и духовных ценностей на более высокий и принципиально иной уровень. Таким образом, изучение современных информационных технологий позволяет человеку легко адаптироваться к современным тенденциям и направлениям развития общества, производственным процессам и повышать уровень социальной жизни.

Обучение всегда начинается с освоения фундаментальных элементов, таких как методы организации, хранения и обмена информацией. В ходе практики были изучены основные концепции и алгоритмы, такие как пузырьковая сортировка, и реализованы методы работы с файлами.

В качестве инструмента разработки были выбраны Microsoft Visual Studio и язык программирования Python. Microsoft Visual Studio представляет собой интегрированную среду разработки приложений для операционной системы Windows, как консольных, так и с графическим интерфейсом пользователя. Она служит платформой для написания, отладки и сборки кода, а также для последующей публикации приложений. В дополнение к стандартным редактору и отладчику, Visual Studio включает в себя компиляторы, инструменты автоматического завершения кода, визуальные конструкторы и множество других компонентов и функций, облегчающих процесс разработки, что делает Visual Studio одной из самых удобных и интуитивно понятных сред разработки.

Язык Python является универсальным языком программирования, который сам по себе не связан с какой-либо операционной системой и с успехом использовался и используется для написания программ и программ обработки текстов и баз данных.

# Постановка задачи

# Требуется разработать и представить функционирующий алгоритм сортировки методом «пузырька». Итоговая программная реализация должна включать в себя непосредственный алгоритм сортировки, а также процедуры для получения входных данных из указанного файла и сохранения отсортированных результатов в другой файл.

### **1.1 Принцип работы**

Принцип работы пузырьковой сортировки (Bubble Sort)

Алгоритм пузырьковой сортировки последовательно сравнивает соседние элементы массива и меняет их местами, если они находятся в неправильном порядке. Процесс повторяется до тех пор, пока весь массив не будет отсортирован. Название происходит от аналогии с "всплытием" крупных элементов (как пузырьков воздуха в воде) в конец массива.

**Исходный массив**: [5, 3, 8, 4, 2]

1. **Первый проход**:
   * [3, 5, 8, 4, 2] → 5 > 3 (меняем).
   * [3, 5, 4, 8, 2] → 8 > 4 (меняем).
   * [3, 5, 4, 2, 8] → 8 > 2 (меняем).  
     Результат: *[3, 5, 4, 2, 8]* (8 "всплыл" в конец).
2. **Второй проход**:
   * [3, 4, 5, 2, 8] → 5 > 4 (меняем).
   * [3, 4, 2, 5, 8] → 5 > 2 (меняем).  
     Результат: *[3, 4, 2, 5, 8]* (5 занял своё место).
3. **Третий проход**:
   * [3, 2, 4, 5, 8] → 4 > 2 (меняем).  
     Результат: *[3, 2, 4, 5, 8]* (4 на месте).
4. **Четвертый проход**:
   * [2, 3, 4, 5, 8] → 3 > 2 (меняем).  
     Итог: *[2, 3, 4, 5, 8]* (массив отсортирован).

### **1.2 Достоинства алгоритма**

Сортировка методом "пузырька" представляет собой один из наиболее элементарных алгоритмов сортировки, отличающийся простотой понимания и реализации. Благодаря своей незамысловатой структуре, он легко воспроизводится практически на любом языке программирования, что делает его идеальным выбором для начинающих программистов. Простота алгоритма "пузырька" делает его хорошей отправной точкой для понимания более сложных алгоритмов сортировки, поскольку принципы, лежащие в его основе, используются в других, более продвинутых методах.

### **1.3 Недостатки алгоритма**

В силу своей невысокой производительности, данный алгоритм чаще рассматривается в образовательных целях и редко находит применение в реальных задачах, особенно при работе с крупными объемами данных. Основной причиной этого является значительное число итераций и операций сравнения, требуемых на каждой итерации, что существенно увеличивает время обработки. Временная сложность алгоритма, выраженная как O(N^2), где N – количество элементов, определяет его как один из наименее эффективных методов сортировки по времени выполнения.

### **1.4 Типичные сценарии использования**

Этот способ упорядочивания данных целесообразно применять в образовательных задачах, особенно полезно для начинающих разработчиков в целях обучения. Он также эффективен при работе с ограниченным объемом входных данных. Данный алгоритм может быть использован для сортировки с целью оптимизации поиска определенных элементов и для приведения данных к определенному порядку.

Кроме того, рассматриваемый метод сортировки демонстрирует свою применимость в ситуациях, когда важна наглядность и простота реализации, а не максимальная производительность.

# 2. Выбор решения

В качестве среды разработки была выбрана Microsoft Visual Studio 2022 с установленным расширением Python Tools for Visual Studio (PTVS), а язык программирования — Python. Python является высокоуровневым языком с динамической типизацией и автоматическим управлением памятью. Он предоставляет мощные структуры данных (списки, словари, множества) и поддерживает различные парадигмы программирования, что делает его идеальным для быстрой реализации алгоритмов, включая сортировку данных.

Для отладки использовались встроенные возможности Visual Studio:

* Пошаговое выполнение (Step Into/Over/Out)
* Точки останова (Breakpoints)
* Инспектор переменных (Watch/Locals Windows)
* Интегрированный отладчик Python

Вся логика приложения реализована в одном файле (PythonApplication2.py). Пользовательский интерфейс выполнен в виде консольного меню (CLI), позволяющего выбирать действия через текстовые команды.

# 3. Описание программы

В рамках представленной работы, студент Кукушкин А.А. занимался выполнением практического задания, посвященного реализации алгоритма «Сортировка пузырьком». В ходе выполнения данного задания, студентом было произведено считывание информации из файла, отображение массива на экране, отображение времени сортировки, а также разработан пользовательский интерфейс для взаимодействия с программой.

Чтение массива из текстового файла осуществляется в функции **load\_array**:

def load\_array(self):

filename = filedialog.askopenfilename(filetypes=[("Text files", "\*.txt")])

if not filename:

return

try:

with open(filename, "r", encoding="utf-8") as f:

size = int(f.readline())

self.arr = list(map(int, f.read().split()))

self.repeat\_sort = False

self.update\_array\_display()

messagebox.showinfo("Успех", "Массив успешно загружен!")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось прочитать файл:\n{e}")

Подробный алгоритм выполнения этой функции представлен в пункте 4 (Рисунок 1).

Отображение текущего массива в отдельном окне осуществляется через функцию **update\_array\_display**:

def update\_array\_display(self):

self.text\_array.delete("1.0", tk.END)

if not self.arr:

self.text\_array.insert(tk.END, "[пусто]")

else:

self.text\_array.insert(tk.END, "\n".join(f"{i + 1}) {val}" for i, val in enumerate(self.arr)))

Подробный алгоритм выполнения этой функции представлен в пункте 4 (Рисунок 2).

Отображение времени сортировки массива в диалоговом окне осуществляется через функцию **show\_time**:

def show\_time(self):

if self.time\_spent == 0.0:

messagebox.showinfo("Время сортировки", "Сначала отсортируйте массив!")

else:

messagebox.showinfo("Время сортировки", f"{self.time\_spent:.6f} сек.")

Подробный алгоритм выполнения этой функции представлен в пункте 4 (Рисунок 3).

Пользовательский интерфейс оформлен с использованием TKinterModernThemes и темой "sun-valley" в темном режиме. Интерфейс включает:

* Ввод размера массива и границ значений;
* Кнопки управления генерацией, сортировкой, загрузкой, сохранением;
* Текстовое поле с прокруткой, в котором отображается массив.

def \_\_init\_\_(self):

* Вызов super().\_\_init\_\_() инициализирует главное окно программы с заголовком "Сортировка пузырьком", установленной темой "sun-valley" и тёмным режимом (mode="dark").
* self.arr — список для хранения элементов массива.
* self.time\_spent — переменная для хранения времени сортировки.
* self.repeat\_sort — флаг, указывающий, был ли массив уже отсортирован.
* StringVar() используется для привязки текстовых значений к полям ввода: size\_var — размер массива, min\_var и max\_var — минимальное и максимальное значения.

super().\_\_init\_\_("Сортировка пузырьком", theme="sun-valley", mode="dark")

self.arr = []

self.time\_spent = 0.0

self.repeat\_sort = False

self.size\_var = tk.StringVar()

self.min\_var = tk.StringVar(value="0")

self.max\_var = tk.StringVar(value="1000000")

Создание фрейма ввода параметров массива.

Фрейм input\_frame служит для ввода параметров массива пользователем.

С помощью методов Label и Entry размещаются поля ввода и соответствующие им подписи.

Кнопка Сгенерировать вызывает метод generate\_array, отвечающий за создание массива с заданными параметрами.

Указание координат row и col определяет расположение виджетов по сетке.

self.input\_frame = self.addFrame("Ввод массива")

self.input\_frame.Label("Размер массива:", row=0, col=0)

self.input\_frame.Entry(textvariable=self.size\_var, row=1, col=0)

self.input\_frame.Label("Минимум:", row=0, col=1)

self.input\_frame.Entry(textvariable=self.min\_var, row=1, col=1)

self.input\_frame.Label("Максимум:", row=0, col=2)

self.input\_frame.Entry(textvariable=self.max\_var, row=1, col=2)

self.input\_frame.Button("Сгенерировать", self.generate\_array, row=2, col=0, colspan=3)

gridkwargs задаёт отступы и расширение фрейма по ширине.

grid\_columnconfigure делает первый столбец растягиваемым при изменении размера окна.

self.input\_frame.gridkwargs = {"padx": 10, "pady": 10, "sticky": "ew"}

self.input\_frame.master.grid\_columnconfigure(0, weight=1)

Фрейм buttons\_frame содержит управляющие кнопки:

1. Сортировать — запускает пузырьковую сортировку массива.
2. Сохранить — записывает массив в текстовый файл.
3. Загрузить — загружает массив из файла.
4. Из URL — загружает массив по ссылке.
5. На GitHub — выгружает массив в файл на GitHub.
6. Время — отображает время последней сортировки.

* Все кнопки размещаются на одной строке (row=0) в разных столбцах

self.buttons\_frame = self.addLabelFrame("Управление")

self.buttons\_frame.Button("Сортировать", self.bubble\_sort, row=0, col=0)

self.buttons\_frame.Button("Сохранить", self.save\_array, row=0, col=1)

self.buttons\_frame.Button("Загрузить", self.load\_array, row=0, col=2)

self.buttons\_frame.Button("Из URL", self.load\_array\_from\_url, row=0, col=3)

self.buttons\_frame.Button("На GitHub", self.upload\_to\_github, row=0, col=4)

self.buttons\_frame.Button("Время", self.show\_time, row=0, col=5)

Отключает возможность ручного изменения размеров главного окна пользователем.

self.root.resizable(False, False)

array\_frame — отдельный фрейм для вывода массива.

tk.Scrollbar добавляет вертикальную прокрутку. Цвета подобраны под тёмную тему интерфейса

self.array\_frame = self.addLabelFrame("Массив")

scrollbar = tk.Scrollbar(self.array\_frame.master, bg="#1e1e1e", troughcolor="#2c2c2c", activebackground="#3a3a3a", highlightbackground="#1e1e1e")

scrollbar.pack(side="right", fill="y")

* tk.Text — текстовое поле, в которое выводится содержимое массива.
* Параметр yscrollcommand связывает поле с полосой прокрутки.
* Метод config(command=...) устанавливает обратную связь между прокруткой и текстовым полем.

self.text\_array = tk.Text(self.array\_frame.master, height=15, yscrollcommand=scrollbar.set)

self.text\_array.pack(fill="both", expand=True, padx=5, pady=5)

scrollbar.config(command=self.text\_array.yview)

# 4. Схема функций

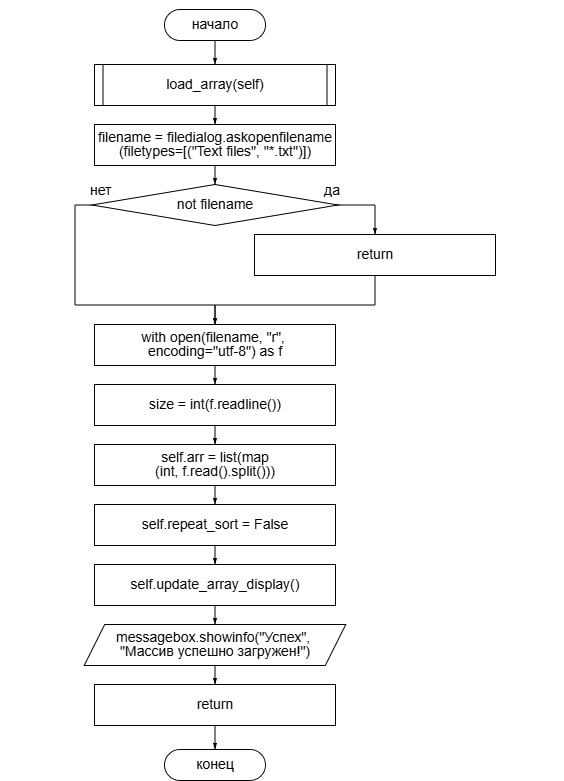


Рисунок 1 - Блок-схема функции *load\_array*

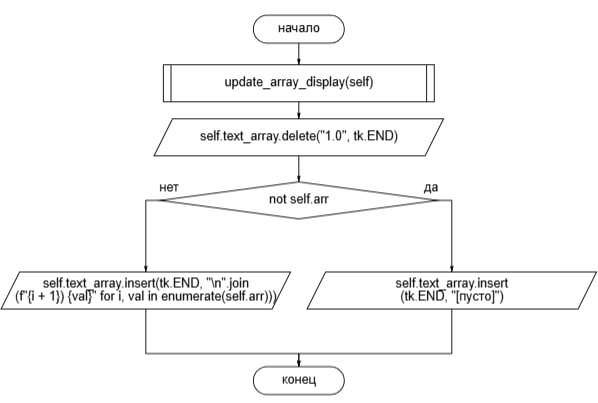


Рисунок 2 - Блок-схема функции *update\_array\_display*

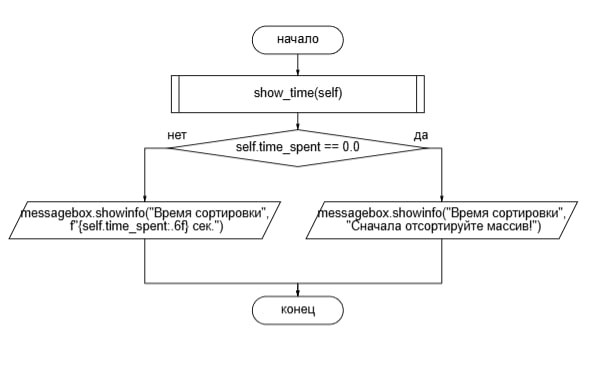


Рисунок 3 - Блок-схема функции *show\_time*

# 5. Отладка

В качестве инструмента для разработки программного обеспечения была задействована интегрированная среда Microsoft Visual Studio, предоставляющая полный спектр необходимых инструментов для создания и отладки как отдельных модулей, так и целых программных комплексов.

Процесс отладки осуществлялся с использованием механизмов точек останова и пошагового выполнения программного кода, что позволяло тщательно анализировать содержимое глобальных и локальных переменных в процессе работы программы.

Тестирование осуществлялось на протяжении всего цикла разработки, начиная с ранних этапов и заканчивая финальной стадией после завершения написания кода. По завершении разработки, специалистом, ответственным за тестирование, были обнаружены и устранены имеющиеся ошибки в программном коде.

# 6. Совместная разработка

В ходе выполнения указанной практической работы, наша команда проводила совместное взаимодействие посредством платформы GitHub.

В GitHub создали проект, где наш магистр выдавал задачи и сроки в которые наша группа должна была выполнить ряд заданий

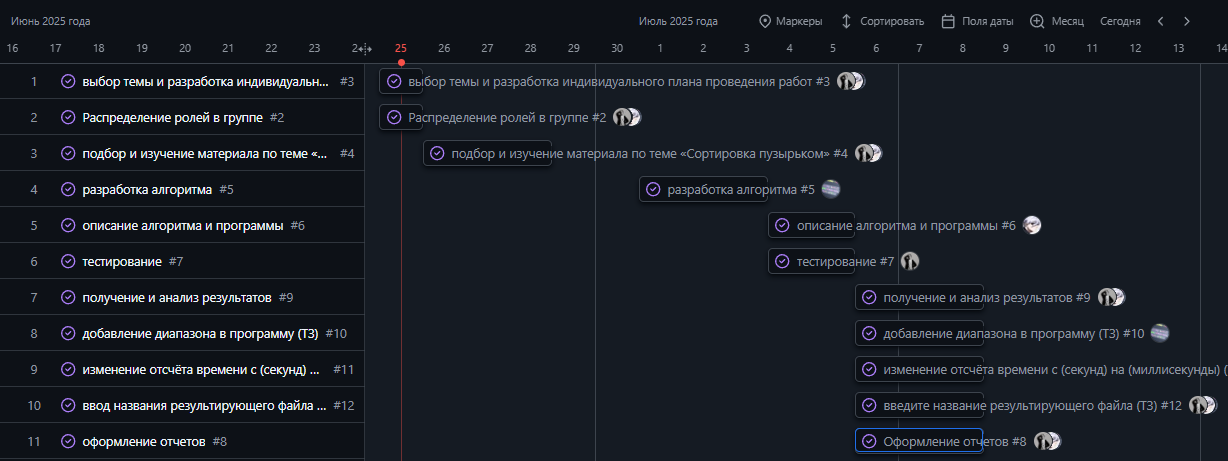


Рисунок 4 - дорожная карта действий группы

Исходный алгоритм был разработан Любченко В.К.

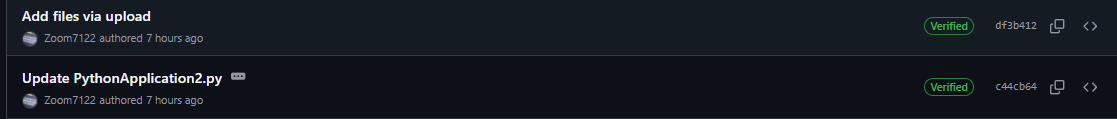
После завершения процесса кодирования, разработанная программа была отправлена в удаленный репозиторий GitHub, в основную "main".

Рисунок 5 – Любченко В.К. выложил выполненное задание на GitHub

Впоследствии, Кукушкин А.А., выступающий в роли второго участника, осуществил загрузку упомянутой программной разработки на свой персональный компьютер. Для этой цели он воспользовался командой **git clone** с указанием соответствующей ссылки. В дальнейшем, им были внесены дополнения, касающиеся пользовательского интерфейса и обновления массива данных. Кроме того, была проведена оптимизация программного кода с целью повышения его эффективности.

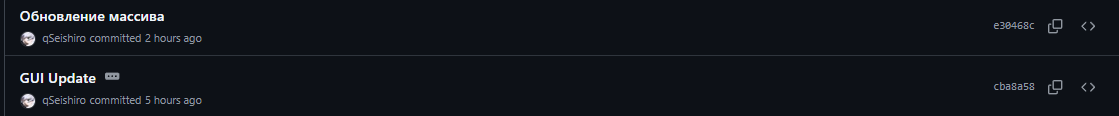


Рисунок 6 – Загрузка доработанного кода на GitHub

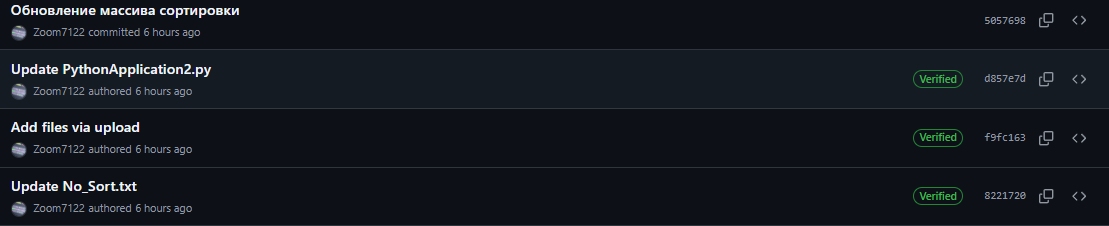
Затем, Любченко В.К. внес коррективы в незначительные ошибки в программном коде и интегрировал процедуру проверки.

Рисунок 7 - Исправление мелких недочетов

Впоследствии, участник под номером три, а именно Комиссаров А.Г., осуществил проверку работоспособности программного обеспечения и разработал соответствующие блок-схемы. Все скриншоты загрузил в отдельные папки в ветке main

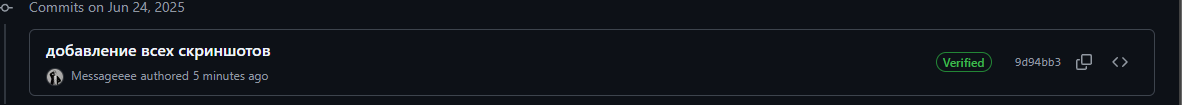


Рисунок 8 – загрузка скриншотов на GitHub

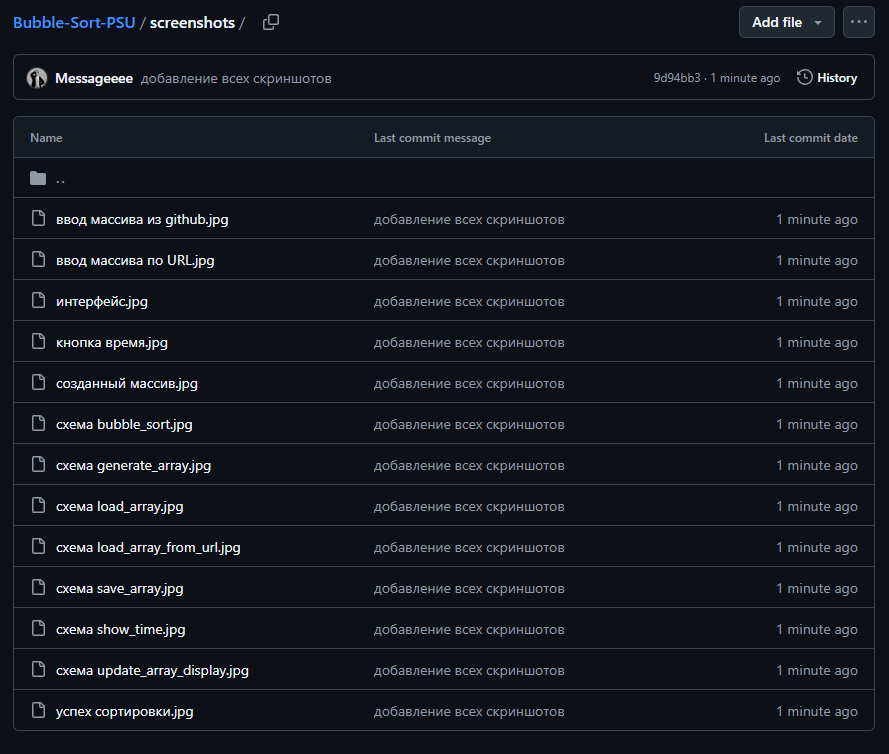


Рисунок 9 – скриншоты на GitHub

**Ссылка на удалённый репозиторий - https://github.com/qSeishiro/Bubble-Sort-PSU.git**

# Заключение

В рамках настоящего учебного проекта были освоены основополагающие принципы и ключевые понятия, относящиеся к алгоритму сортировки "пузырьком", а также реализована система обмена данными с файлами. На практике была продемонстрирована работоспособность алгоритма сортировки методом "пузырька". Разработанная программа осуществляет не только непосредственную реализацию алгоритма, но и обеспечивает возможность считывания исходной информации из файла, а также сохранения полученных результатов в файл.

В процессе отладки было проведено тестирование программного кода, которое подтвердило правильность функционирования алгоритма. Предусмотрены ограничения, касающиеся формата входных числовых данных, и реализована обработка исключительных ситуаций, возникающих при чтении файлов. Результаты проведенного тестирования указывают на рациональность применения данного алгоритма для сортировки массивов, характеризующихся небольшим объемом. Зафиксировано экспоненциальное увеличение времени выполнения программы при увеличении числа элементов, подлежащих сортировке.

Сортировка "пузырьком" представляет собой сравнительно простой и интуитивно понятный алгоритм, который демонстрирует эффективность при работе с небольшими числовыми последовательностями, частично упорядоченными наборами данных, а также в контексте образовательных задач. Несмотря на простоту реализации, значительная вычислительная сложность и существенные временные затраты, необходимые для выполнения сортировки, существенно ограничивают сферу практического использования данного метода.

# Список использованных источников

1. Официальная документация Python [Электронный ресурс] -  [https://docs.python.org/3/](https://docs.python.org/3/" \t "https://chat.deepseek.com/a/chat/s/_blank)
2. Документация Tkinter [Электронный ресурс] -  [https://docs.python.org/3/library/tkinter.html](https://docs.python.org/3/library/tkinter.html" \t "https://chat.deepseek.com/a/chat/s/_blank)
3. GitHub REST API Documentation [Электронный ресурс] -   [https://docs.github.com/en/rest](https://docs.github.com/en/rest" \t "https://chat.deepseek.com/a/chat/s/_blank)
4. Random Number Generation [Электронный ресурс] -  [https://docs.python.org/3/library/random.html](https://docs.python.org/3/library/random.html" \t "https://chat.deepseek.com/a/chat/s/_blank)

**Приложение А**

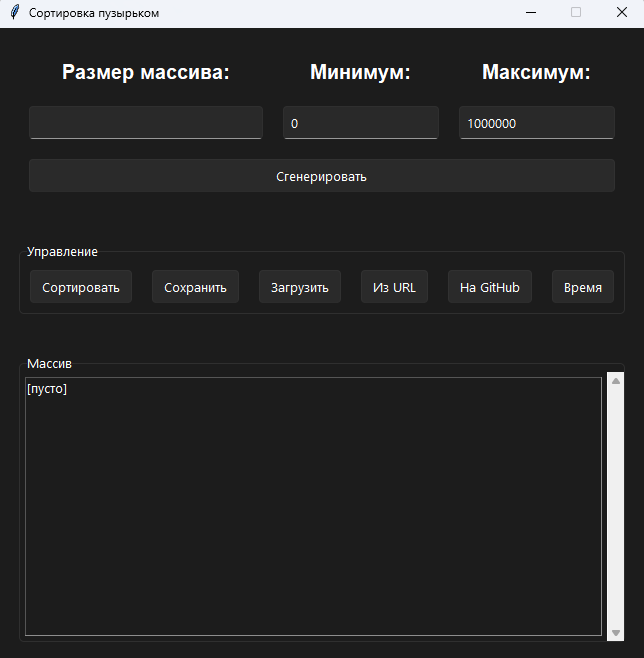


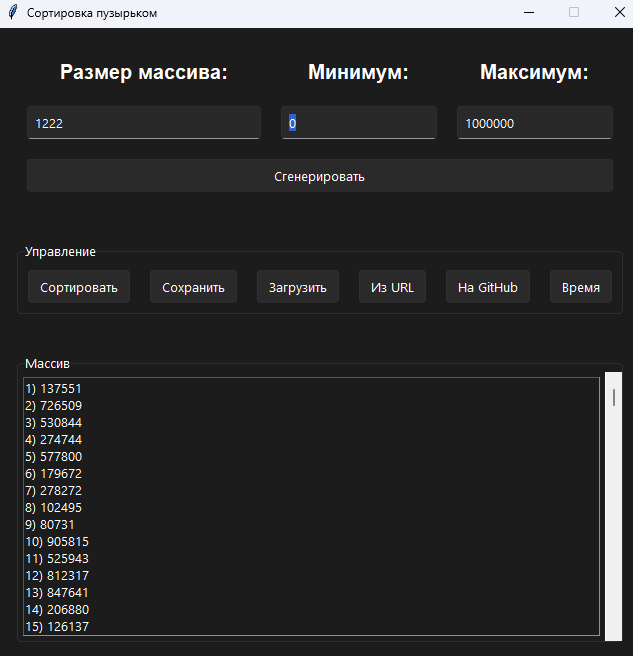
Рисунок А.1 - главный экран

Рисунок А.2 - Созданный массив

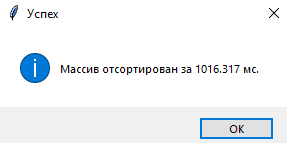


Рисунок А.3 - Успех сортировки

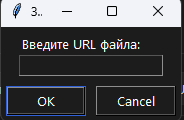


Рисунок А.4 - ввод массива по URL

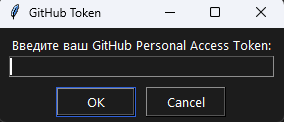


Рисунок А.5 - сохранение массива на github

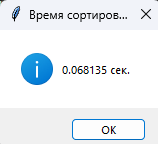


Рисунок А.6 - Кнопка “Время”

# Приложение Б. Листинг программы.

**Файл PythonApplication2.py**

import TKinterModernThemes as TKMT

import tkinter as tk

from tkinter import filedialog, messagebox, simpledialog

import random

import time

import urllib.request

import urllib.error

import base64

import json

class App(TKMT.ThemedTKinterFrame):

def generate\_array(self):

try:

size = int(self.size\_var.get())

min\_val = int(self.min\_var.get())

max\_val = int(self.max\_var.get())

if size <= 0:

raise ValueError("Размер должен быть положительным")

if min\_val > max\_val:

min\_val, max\_val = max\_val, min\_val

self.arr = [random.randint(min\_val, max\_val) for \_ in range(size)]

self.repeat\_sort = False

self.update\_array\_display()

self.time\_spent = 0.0

except ValueError:

messagebox.showerror("Ошибка", "Введите корректные числовые значения!")

def load\_array\_from\_url(self):

url = simpledialog.askstring("Загрузка по URL", "Введите URL файла:")

if not url:

return

try:

with urllib.request.urlopen(url) as response:

content = response.read().decode('utf-8').strip()

lines = content.split('\n')

if len(lines) > 1:

try:

size = int(lines[0])

self.arr = list(map(int, lines[1].split()))

except:

self.arr = []

for line in lines:

self.arr.extend(map(int, line.split()))

else:

self.arr = list(map(int, lines[0].split()))

self.update\_array\_display()

self.repeat\_sort = False

messagebox.showinfo("Успех", f"Успешно загружено {len(self.arr)} элементов по URL!")

except Exception as e:

messagebox.showerror("Ошибка", f"Ошибка при загрузке файла:\n{str(e)}")

def upload\_to\_github(self):

if not self.arr:

messagebox.showwarning("Ошибка", "Сначала создайте массив!")

return

token = simpledialog.askstring("GitHub Token", "Введите ваш GitHub Personal Access Token:", show='\*')

if not token:

return

try:

check\_req = urllib.request.Request(

"https://api.github.com/user",

headers={"Authorization": f"token {token}", "User-Agent": "PythonApp"}

)

with urllib.request.urlopen(check\_req) as response:

user\_data = json.loads(response.read().decode())

scopes = response.headers.get('X-OAuth-Scopes', '')

if 'repo' not in scopes:

messagebox.showerror("Ошибка прав", "Токен не имеет права 'repo'!")

return

except urllib.error.HTTPError as e:

error\_msg = e.read().decode()

messagebox.showerror("Ошибка токена", f"Ошибка проверки токена ({e.code}):\n{error\_msg}")

return

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось проверить токен: {str(e)}")

return

repo\_owner = simpledialog.askstring("Репозиторий", "Владелец репозитория:")

if not repo\_owner:

return

repo\_name = simpledialog.askstring("Репозиторий", "Имя репозитория:")

if not repo\_name:

return

try:

repo\_req = urllib.request.Request(

f"https://api.github.com/repos/{repo\_owner}/{repo\_name}",

headers={"Authorization": f"token {token}", "User-Agent": "PythonApp"}

)

with urllib.request.urlopen(repo\_req) as response:

repo\_data = json.loads(response.read().decode())

permissions = repo\_data.get('permissions', {})

if not permissions.get('push', False):

messagebox.showerror("Ошибка доступа", "Нет прав на запись в этот репозиторий!")

return

except urllib.error.HTTPError as e:

if e.code == 404:

messagebox.showerror("Ошибка", "Репозиторий не найден!")

else:

messagebox.showerror("Ошибка", f"HTTP ошибка {e.code}")

return

file\_path = simpledialog.askstring("Файл", "Путь к файлу в репозитории:", initialvalue="array.txt")

if not file\_path:

return

commit\_message = simpledialog.askstring("Коммит", "Сообщение коммита:", initialvalue="Обновление массива")

if not commit\_message:

return

try:

content = f"{len(self.arr)}\n" + " ".join(map(str, self.arr))

content\_bytes = content.encode("utf-8")

content\_base64 = base64.b64encode(content\_bytes).decode("utf-8")

url = f"https://api.github.com/repos/{repo\_owner}/{repo\_name}/contents/{file\_path}"

data = {

"message": commit\_message,

"content": content\_base64,

"branch": "main"

}

sha = None

try:

req = urllib.request.Request(

url,

headers={"Authorization": f"token {token}", "User-Agent": "PythonApp"}

)

with urllib.request.urlopen(req) as response:

existing\_data = json.loads(response.read().decode())

sha = existing\_data.get("sha")

except urllib.error.HTTPError as e:

if e.code != 404:

error\_msg = e.read().decode()

messagebox.showerror("Ошибка", f"Ошибка проверки файла ({e.code}):\n{error\_msg}")

return

if sha:

data["sha"] = sha

req = urllib.request.Request(

url,

data=json.dumps(data).encode("utf-8"),

headers={"Authorization": f"token {token}", "User-Agent": "PythonApp", "Content-Type": "application/json"},

method="PUT"

)

with urllib.request.urlopen(req) as response:

if response.status in (200, 201):

messagebox.showinfo("Успех", "Файл успешно загружен на GitHub!")

else:

messagebox.showerror("Ошибка", "Ошибка при загрузке файла")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось загрузить файл на GitHub:\n{str(e)}")

def bubble\_sort(self):

if self.repeat\_sort:

messagebox.showinfo("Повтор", "Массив уже отсортирован")

return

if not self.arr:

messagebox.showwarning("Ошибка", "Сначала создайте массив!")

return

start = time.time()

for i in range(len(self.arr) - 1):

swapped = False

for j in range(len(self.arr) - i - 1):

if self.arr[j] > self.arr[j + 1]:

self.arr[j], self.arr[j + 1] = self.arr[j + 1], self.arr[j]

swapped = True

if not swapped:

break

end = time.time()

self.time\_spent = end - start

self.repeat\_sort = True

self.update\_array\_display()

messagebox.showinfo("Успех", f"Массив отсортирован за {self.time\_spent:.20f} сек.")

def save\_array(self):

if not self.arr:

messagebox.showwarning("Ошибка", "Сначала создайте массив!")

return

filename = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "\*.txt")])

if not filename:

return

try:

with open(filename, "w", encoding="utf-8") as f:

f.write(f"{len(self.arr)}\n")

f.write(" ".join(map(str, self.arr)))

messagebox.showinfo("Успех", "Массив успешно записан!")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось сохранить файл:\n{e}")

filedialog.askopenfilename(filetypes=[("Text files", "\*.txt")])

if not filename:

return

try:

with open(filename, "r", encoding="utf-8") as f:

size = int(f.readline())

self.arr = list(map(int, f.read().split()))

self.repeat\_sort = False

self.update\_array\_display()

messagebox.showinfo("Успех", "Массив успешно загружен!")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось прочитать файл:\n{e}")

def update\_array\_display(self):

self.text\_array.delete("1.0", tk.END)

if not self.arr:

self.text\_array.insert(tk.END, "[пусто]")

else:

self.text\_array.insert(tk.END, "\n".join(f"{i + 1}) {val}" for i, val in enumerate(self.arr)))

def show\_time(self):

if self.time\_spent == 0.0:

messagebox.showinfo("Время сортировки", "Сначала отсортируйте массив!")

else:

messagebox.showinfo("Время сортировки", f"{self.time\_spent:.6f} сек.")

def \_\_init\_\_(self):

super().\_\_init\_\_("Сортировка пузырьком", theme="sun-valley", mode="dark")

self.arr = []

self.time\_spent = 0.0

self.repeat\_sort = False

self.size\_var = tk.StringVar()

self.min\_var = tk.StringVar(value="0")

self.max\_var = tk.StringVar(value="1000000")

self.input\_frame = self.addFrame("Ввод массива")

self.input\_frame.Label("Размер массива:", row=0, col=0)

self.input\_frame.Entry(textvariable=self.size\_var, row=1, col=0)

self.input\_frame.Label("Минимум:", row=0, col=1)

self.input\_frame.Entry(textvariable=self.min\_var, row=1, col=1)

self.input\_frame.Label("Максимум:", row=0, col=2)

self.input\_frame.Entry(textvariable=self.max\_var, row=1, col=2)

self.input\_frame.Button("Сгенерировать", self.generate\_array, row=2, col=0, colspan=3)

self.input\_frame.gridkwargs = {"padx": 10, "pady": 10, "sticky": "ew"}

self.input\_frame.master.grid\_columnconfigure(0, weight=1)

self.buttons\_frame = self.addLabelFrame("Управление")

self.buttons\_frame.Button("Сортировать", self.bubble\_sort, row=0, col=0)

self.buttons\_frame.Button("Сохранить", self.save\_array, row=0, col=1)

self.buttons\_frame.Button("Загрузить", self.load\_array, row=0, col=2)

self.buttons\_frame.Button("Из URL", self.load\_array\_from\_url, row=0, col=3)

self.buttons\_frame.Button("На GitHub", self.upload\_to\_github, row=0, col=4)

self.buttons\_frame.Button("Время", self.show\_time, row=0, col=5)

self.root.resizable(False, False)

self.array\_frame = self.addLabelFrame("Массив")

scrollbar = tk.Scrollbar(self.array\_frame.master, bg="#1e1e1e", troughcolor="#2c2c2c", activebackground="#3a3a3a", highlightbackground="#1e1e1e")

scrollbar.pack(side="right", fill="y")

self.text\_array = tk.Text(self.array\_frame.master, height=15, yscrollcommand=scrollbar.set)

self.text\_array.pack(fill="both", expand=True, padx=5, pady=5)

scrollbar.config(command=self.text\_array.yview)

self.update\_array\_display()

self.run()

if \_\_name\_\_ == "\_\_main\_\_":

App()