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)

Комиссаров Андрей Геннадьевич

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 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, Карамышева Н.С.

*(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)

Комиссаров Андрей Геннадьевич

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 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, Карамышева Н.С.

*(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 | 01.07.25 –  03.07.25 |  |
| 4 | Описание алгоритма и программы | 18 | 03.07.25 –  04.07.25 |  |
| 5 | Тестирование | 5 | 04.07.25 –  05.07.25 |  |
| 6 | Получение и анализ результатов | 10 | 05.07.25 –  08.07.25 |  |
| 7 | Оформление отчёта | 15 | 05.07.25 –  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)

Комиссаров Андрей Геннадьевич

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 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

А.Г. Комиссаров реализовал практическую работу, посвященную алгоритму «Сортировка пузырьком». В самом начале он ознакомился и тщательно разобрал принцип работы данного алгоритма. Затем был определен оптимальный способ реализации и выбран язык программирования Python. На этом языке была создана программа, реализующая сортировку массива данных методом пузырька. Кроме того, была выполнена работа с файловой системой. Завершающим этапом стали тестирование и отладка разработанного программного кода. Результаты работы были оформлены в виде отчета.

Бакалавр Комиссаров А.Г. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Руководитель Карамышева Н.С.\_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

Комиссаров Андрей Геннадьевич

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 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

В ходе прохождения практики Комиссаров А.Г. занимался решением ряда задач, среди которых: разработка структурных схем и отладка программного кода.

За время практики были изучены ключевые принципы и методы пузырьковой сортировки, а также освоены приемы работы с файловой системой. В процессе работы Комиссаров А.Г. продемонстрировал себя как ответственный и усердный студент, обладающий хорошим знанием материала, осведомленностью о последних достижениях в науке и владеющий актуальными общенаучными знаниями в области информатики, вычислительной техники, программирования и сортировки данных. По итогам выполненной работы Комиссаров А.Г. достоин оценки «\_\_».

Руководитель практики к.т.н., доцент, Карамышева Н.С. « » 2025 г.

**Содержание**

[**Введение 8**](#_Toc201677127)

[**1 Постановка задачи 10**](#_Toc201677128)

[**1.1 Достоинства алгоритма 10**](#_Toc201677130)

[**1.2 Недостатки алгоритма 10**](#_Toc201677131)

[**1.3 Типичные сценарии использования 10**](#_Toc201677132)

[**2 Выбор решения 12**](#_Toc201677133)

[**3.Схема программы 13**](#_Toc201677134)

[**4.Тестирование программы 18**](#_Toc201677135)

[**4.1 Тестирование интерфейса 18**](#_Toc201677136)

[**4.2 Тестирование на разных наборах данных 21**](#_Toc201677137)

[**4.3Анализ полученных результатов тестирования 21**](#_Toc201677138)

[**5. Совместная разработка 22**](#_Toc201677139)

[**Заключение 22**](#_Toc201677140)

[**Список использованных источников 23**](#_Toc201677141)

[**Приложение A 24**](#_Toc201677142)

[**Приложение Б. Листинг 26**](#_Toc201677143)

# Введение

В наши дни информационные технологии глубоко проникли во все сферы деятельности человека и стали неотъемлемой частью повседневной жизни. Существенное снижение издержек производства, ускорение социальных изменений и, следовательно, повышение общей эффективности связаны с эволюцией возможностей обработки, хранения, передачи и визуализации данных.

Обработка информации нового качества, полученной в результате этих процессов, позволяет создавать и использовать материальные и духовные ценности на качественно ином уровне. Обучение всегда начинается с освоения фундаментальных принципов, таких как методы сортировки, хранения и передачи данных. В процессе прохождения практики я изучил основные концепции и алгоритмы пузырьковой сортировки, а также освоил работу с файловой системой.

В качестве инструмента разработки были выбраны Microsoft Visual Studio и язык программирования Python.

Microsoft Visual Studio представляет собой интегрированную среду разработки (IDE) для создания приложений под управлением операционной системы Windows, как консольных, так и с графическим интерфейсом. Она служит отправной точкой для написания, отладки и сборки кода, а также для последующей публикации готовых приложений. Помимо стандартных инструментов, таких как редактор кода и отладчик, Visual Studio включает в себя компиляторы, средства автоматического завершения кода, графические редакторы и другие функции, упрощающие процесс разработки, что делает эту среду одной из наиболее доступных и удобных в использовании.

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.

# 1 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 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.

### **1.2 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.

### **1.3 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.

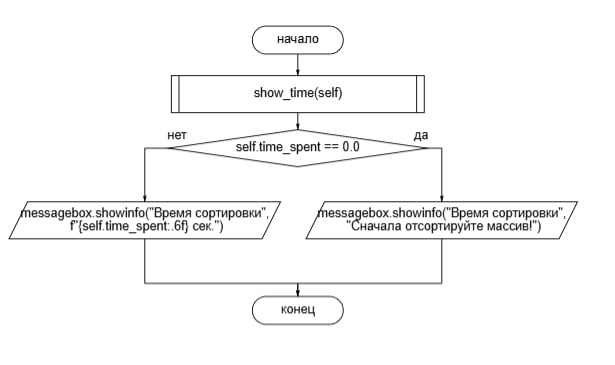
# 2 Выбор решения

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.

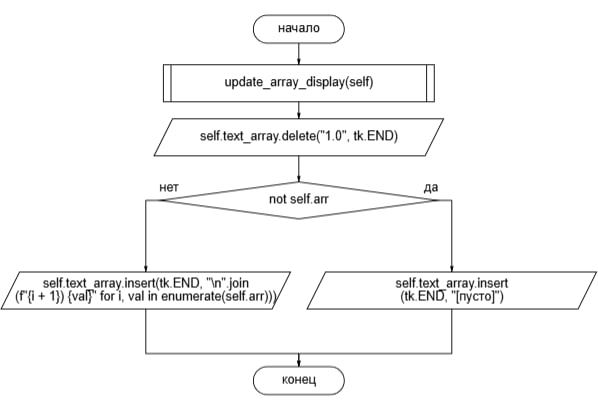
* Для отладки использовались встроенные возможности Visual Studio:
* Пошаговое выполнение (Step Into/Over/Out)
* Точки останова (Breakpoints)
* Инспектор переменных (Watch/Locals Windows)
* Интегрированный отладчик Python

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

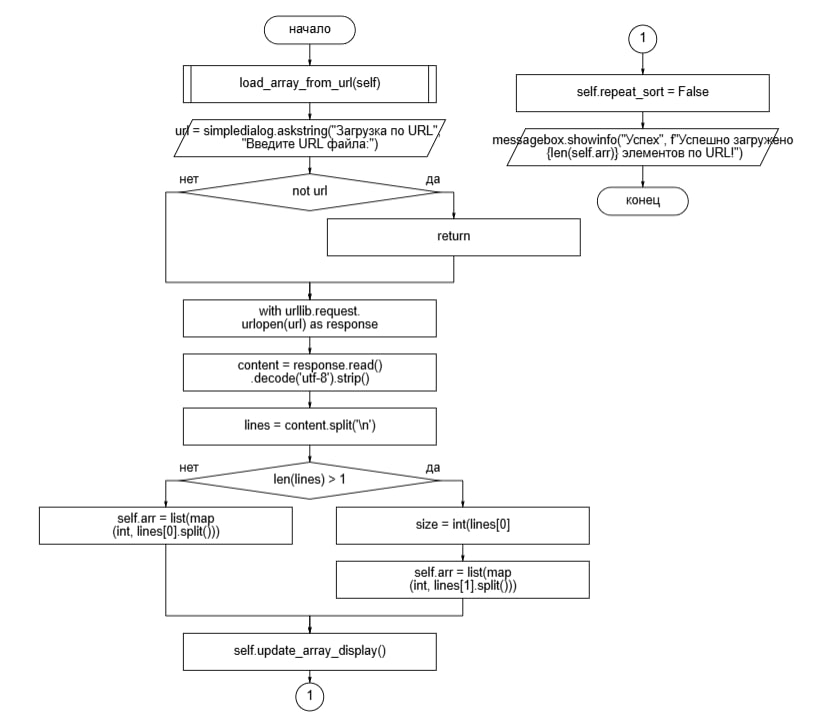
# 3.Схема программы



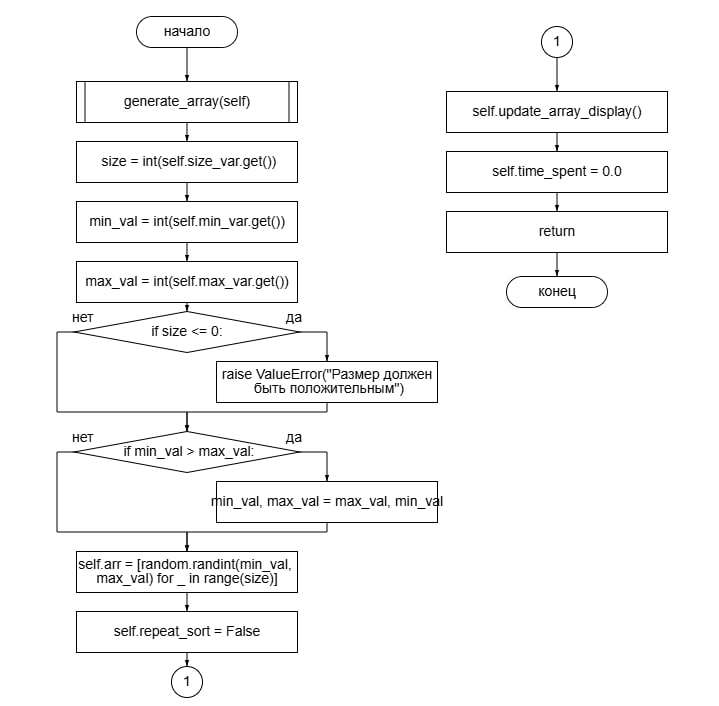
**Рисунок 1 – схема алгоритма show\_time**



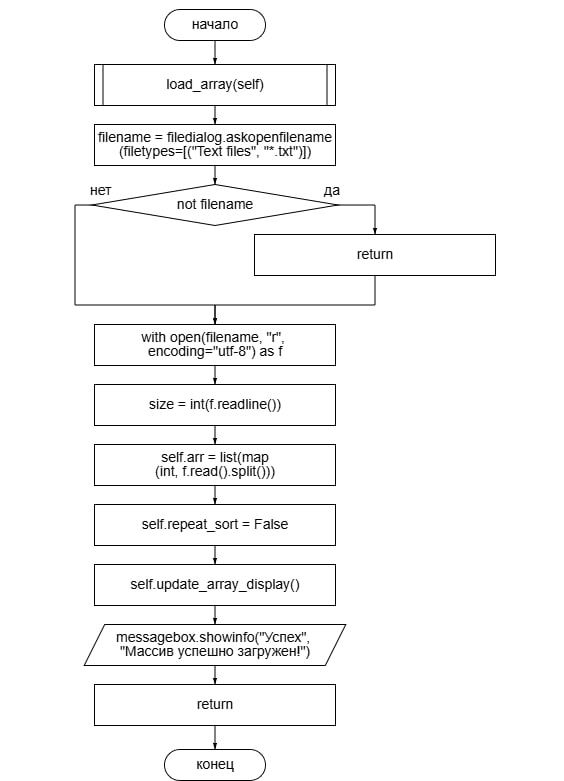
**Рисунок 2 - схема алгоритма update\_array\_display**



**Рисунок 3 - схема алгоритма load\_array\_from\_url**



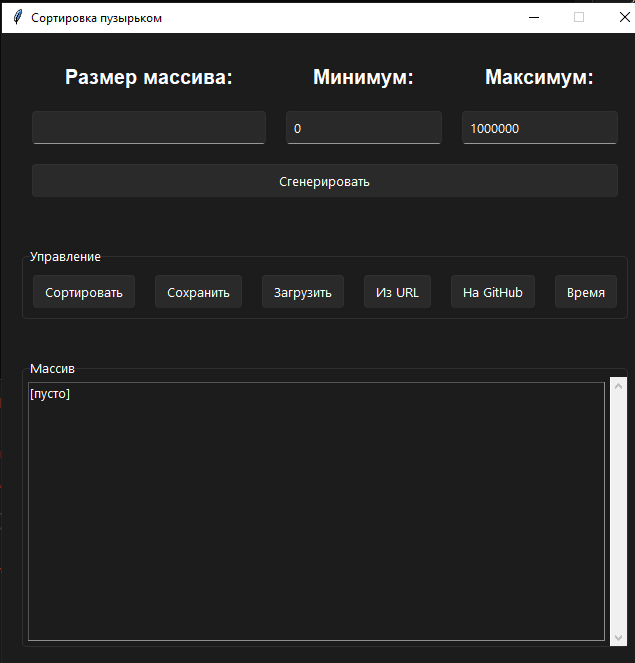
**Рисунок 4 - схема алгоритма generate\_array**



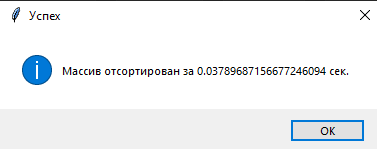
**Рисунок 5 - схема алгоритма load\_array**

# 4.Тестирование программы

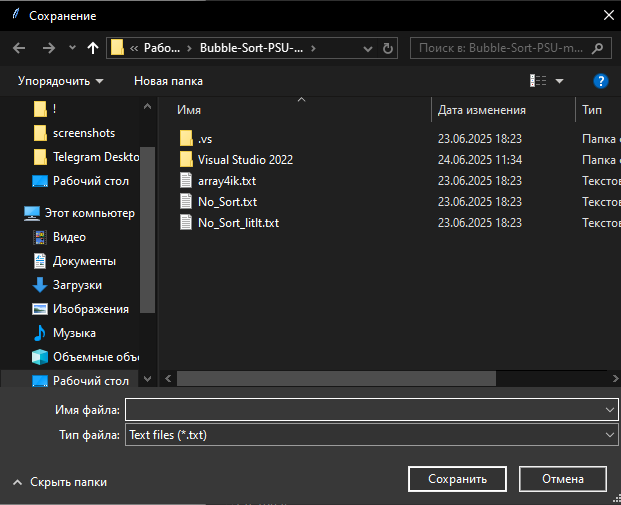
**4.1 Тестирование интерфейса**

****

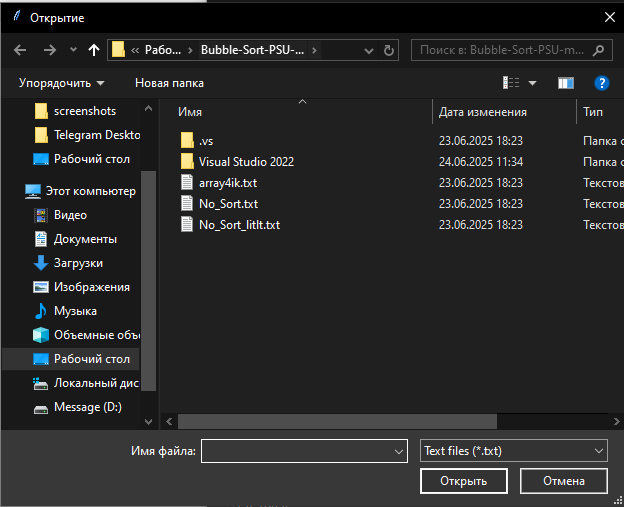
**Рисунок 6 - интерфейс программы**

****

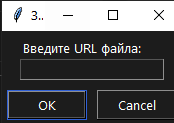
**Рисунок 7 - пункт меню "сортировать"**



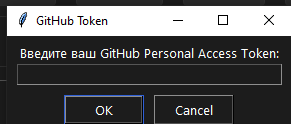
**Рисунок 8 - пункт меню "сохранить”**

****

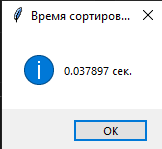
**Рисунок 9 - пункт меню "открыть"**

****

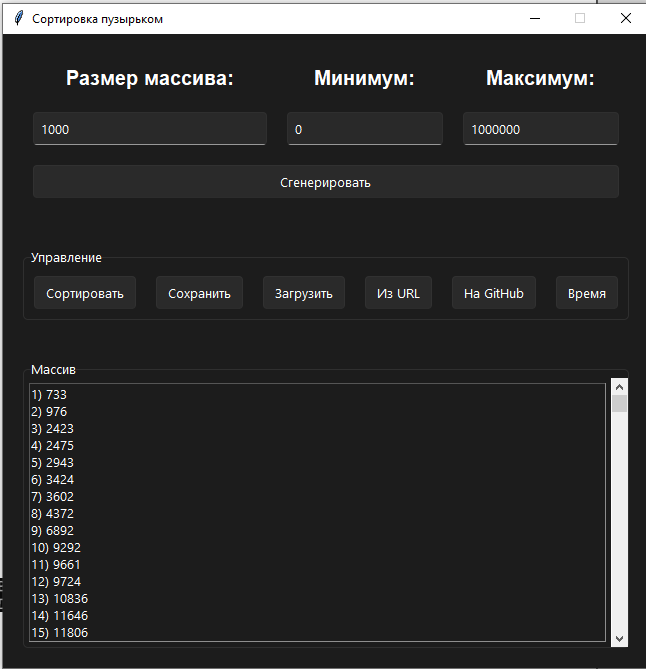
**Рисунок 10 - пункт меню "из URL"**

****

**Рисунок 11 - пункт меню "на GitHub"**

****

**Рисунок 12 - пункт меню "время"**

****

**Рисунок 13 - пункт меню "сгенерировать"**

**4.2 Тестирование на разных наборах данных**

Таблица 1 - Тестовый набор данных

|  |  |  |
| --- | --- | --- |
| № теста | Количество элементов | Время выполнения в  секундах |
| 1 | 5000 | 0.98664 |
| 2 | 6000 | 1.45761 |
| 3 | 7000 | 1.98282 |
| 4 | 8000 | 2.58672 |
| 5 | 9000 | 3.20712 |
| 6 | 10000 | 4.06253 |
| 7 | 11000 | 4.86850 |
| 8 | 12000 | 6.14722 |
| 9 | 13000 | 6.75634 |
| 10 | 14000 | 8.18866 |

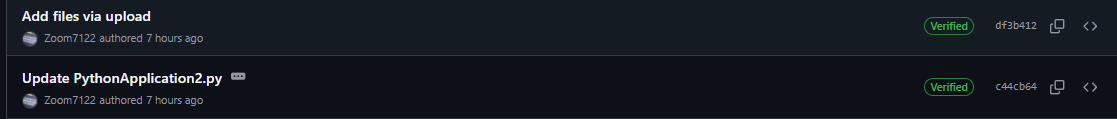
**4.3Анализ полученных результатов тестирования**

В результате анализа функционирования алгоритма было установлено, что с увеличением размерности входной последовательности данных наблюдается экспоненциальный рост времени, затрачиваемого на сортировку. Данная закономерность указывает на то, что рассматриваемый алгоритм обладает низкой производительностью при обработке значительных объемов информации. Таким образом, применение данной сортировки нецелесообразно в условиях работы с большими наборами данных.

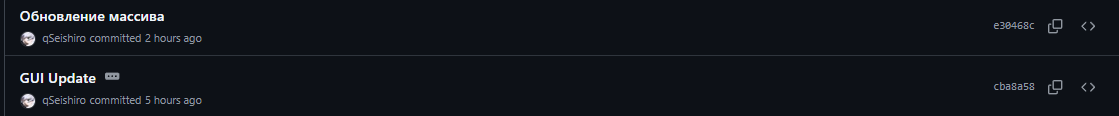
# 5. Совместная разработка

В рамках выполнения указанного практического задания, наша рабочая группа осуществляла взаимодействие посредством платформы GitHub.

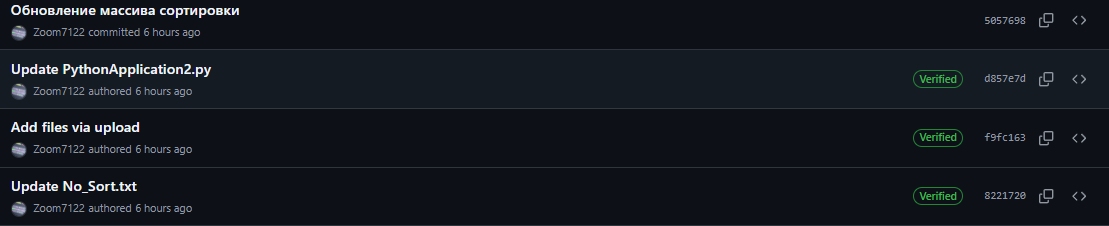
Изначальная версия алгоритма была разработана и представлена Любченко В.К.

По завершении этапа программирования, разработанное приложение было загружено в удаленный репозиторий GitHub, конкретно в основную ветвь, обозначенную как "main".

**Рисунок 7 – Загрузка программы на GitHub**

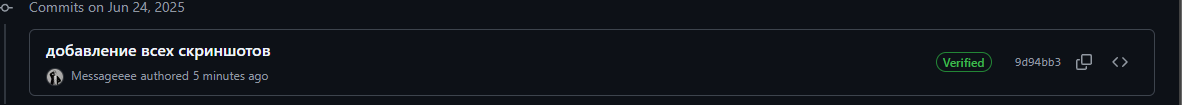
Впоследствии, Кукушкин А.А., являясь вторым участником, осуществил перенос указанного программного обеспечения на свой персональный компьютер. Для этого была использована команда git clone, сопровождаемая соответствующей ссылкой. В ходе дальнейшей работы им были внесены дополнения, касающиеся пользовательского интерфейса и процедуры обновления массива данных. Кроме того, была проведена оптимизация программного кода.

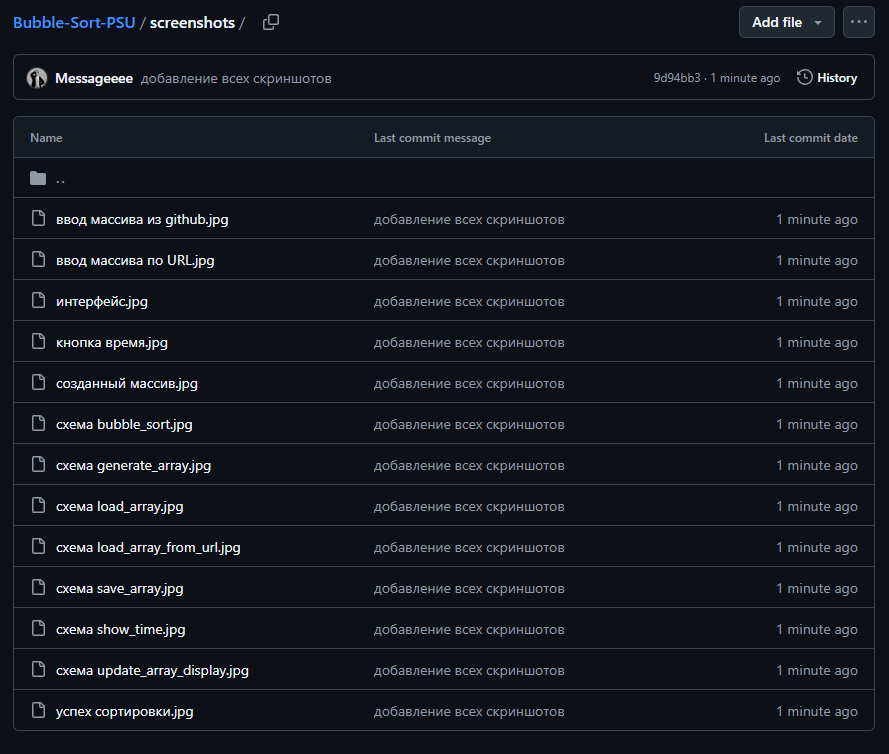
**Рисунок 8 – Загрузка доработанного кода на GitHub**

Впоследствии, Любченко В.К. устранил незначительные ошибки в программном коде и внедрил дополнительный механизм контроля.

**Рисунок 9 – 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 10 – загрузка скриншотов на GitHub**



**Рисунок 11 – скриншоты на GitHub**

# Заключение

В рамках данной учебно-практической работы были изучены ключевые принципы и инструменты алгоритма пузырьковой сортировки, а также разработан механизм взаимодействия с файлами. На практике была продемонстрирована работа сортировки методом "пузырька". Итоговая программа включает в себя реализацию самого алгоритма, а также функционал для чтения исходных данных из файла и записи результатов в файл.

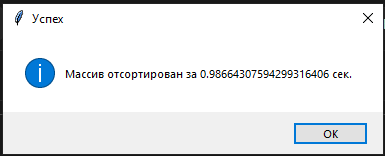
В процессе тестирования была выполнена отладка кода, подтверждена корректность работы алгоритма, учтены ограничения на формат вводимых чисел и обработка ошибок, связанных с чтением файлов.

Результаты тестирования показали, что данный алгоритм целесообразно использовать для сортировки массивов небольшого размера, поскольку с увеличением количества сортируемых элементов время выполнения программы возрастает экспоненциально. Пузырьковая сортировка представляет собой относительно простой и понятный алгоритм, который может быть эффективен при работе с небольшими наборами чисел, частично упорядоченными последовательностями, а также в образовательных целях. Несмотря на простоту реализации, высокая вычислительная сложность и значительные временные затраты на сортировку существенно ограничивают область применения данного метода.

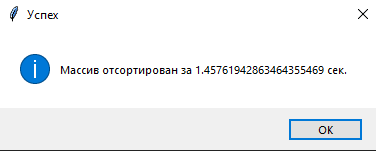
# Список использованных источников

1. Официальная документация Python [Электронный ресурс] -  <https://docs.python.org/3/>
2. Документация Tkinter [Электронный ресурс] -  <https://docs.python.org/3/library/tkinter.html>
3. GitHub REST API Documentation [Электронный ресурс] -   <https://docs.github.com/en/rest>
4. Random Number Generation [Электронный ресурс] -  <https://docs.python.org/3/library/random.html>

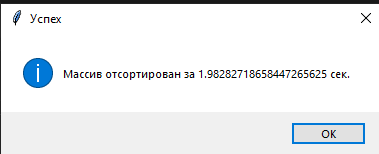
# Приложение A



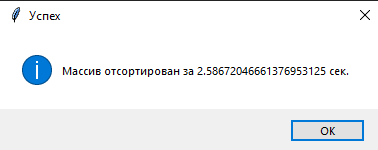
**Рисунок А. 1 -тестирование при 5000 элементов**

****

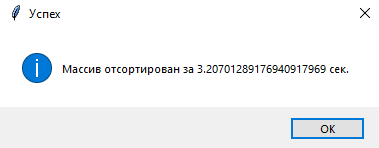
**Рисунок А. 2 - тестирование при 6000 элементов**

****

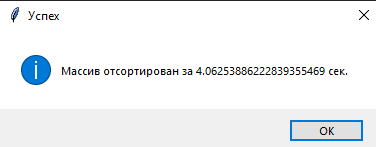
**Рисунок А. 3 - тестирование при 7000 элементов**

****

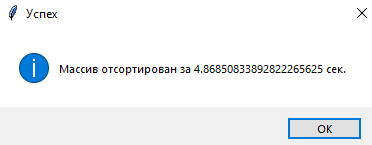
**Рисунок А. 4 - тестирование при 8000 элементов**

****

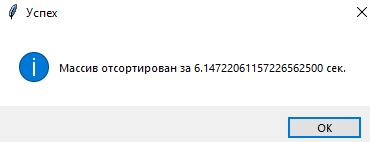
**Рисунок А. 5 - тестирование при 9000 элементов**

****

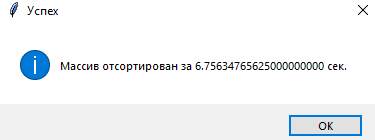
**Рисунок А. 6 - тестирование при 10000 элементов**

****

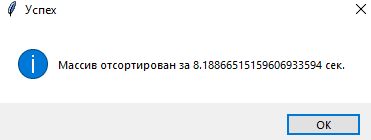
**Рисунок А. 7 - тестирование при 11000 элементов**

****

**Рисунок А. 8 - тестирование при 12000 элементов**

****

**Рисунок А. 9 - тестирование при 13000 элементов**

****

**Рисунок А. 10 - тестирование при 14000 элементов**

# Приложение Б. Листинг

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}")

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}")

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()