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)

Любченко Владислав Константинович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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

Год обучения1семестр 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 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)

Любченко Владислав Константинович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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 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 | 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**

(2025/2025 учебный год)

Любченко Владислав Константинович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

В.К. Любченко завершил практическую работу, посвященную алгоритму сортировки пузырьком. В рамках задания был проведен всесторонний анализ принципов работы алгоритма пузырьковой сортировки. В качестве инструмента реализации был выбран язык программирования Python, на котором была разработана программа, осуществляющая сортировку массива данных посредством метода пузырька. Дополнительно была произведена работа с файловой системой для ввода и вывода данных. Результаты работы были задокументированы в форме отчёта.

Бакалавр Любченко В.К. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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**

(2025/2025 учебный год)

Любченко Владислав Константинович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.2025 по 08.07.2025

Кафедра «Вычислительная техника»

В ходе прохождения практики Любченко В.К. занимался рассмотрением алгоритма сортировки пузырьком, включая изучение принципов его функционирования.

В период практики были изучены ключевые концепции и механизмы пузырьковой сортировки, а также реализован функционал взаимодействия с файловой системой. В процессе выполнения поставленных задач Любченко В.К. продемонстрировал высокий уровень ответственности, усердие в работе и глубокое понимание предметной области. Он показал знание актуальных тенденций в науке, а также владение современными общенаучными знаниями в области информатики, вычислительной техники, программирования и алгоритмов сортировки.

За выполнение работы Любченко В.К. заслуживает оценки «\_\_\_\_\_\_\_\_\_\_».

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

Содержание

[Введение 7](#_Toc139480638)

[1 Постановка задачи 9](#_Toc139480639)

[1.1 Достоинства алгоритма 9](#_Toc139480640)

[1.2 Недостатки алгоритма 9](#_Toc139480641)

[1.3 Типичные сценарии использования 9](#_Toc139480642)

[2 Выбор решения 10](#_Toc139480643)

[3 Описание программы 11](#_Toc139480644)

[4. Схема программы 13](#_Toc139480645)

[5. Отладка 15](#_Toc139480646)

[6. Совместная разработка 16](#_Toc139480647)

[Заключение 17](#_Toc139480648)

[Список использованных источников 18](#_Toc139480649)

[Приложение А 18](#_Toc139480650)

[Приложение Б. Листинг 19](#_Toc139480651)

# Введение

В нынешнюю эпоху информационные технологии глубоко проникли в повседневную жизнь подавляющего большинства людей. Сокращение затрат на производство, ускорение общественных изменений и, как следствие, повышение их продуктивности в настоящее время неразрывно связаны с кардинальными изменениями в возможностях сбора, анализа, хранения, распространения и визуализации данных, произошедшими за последние десятилетия.

Получение информации нового качественного уровня благодаря этим процессам позволяет поднять процесс создания и использования материальных и духовных ценностей на качественно новый уровень. Таким образом, изучение современных информационных технологий позволяет человеку легко ориентироваться в современных направлениях и путях развития общества, производственных процессах и повышать качество жизни в обществе.

Обучение всегда начинается с изучения основных элементов, таких как методы упорядочивания, сохранения и обмена данными. В ходе практики я освоил основные принципы и технологии сортировки методом пузырька, реализовал метод работы с файлами.

В качестве инструмента для реализации были выбраны Microsoft Visual Studio и язык программирования Python. Microsoft Visual Studio представляет собой программную среду разработки приложений для 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. Помимо стандартных редактора и отладчика, Visual Studio включает компиляторы, инструменты автоматического завершения кода, графические конструкторы и многие другие элементы и функции, облегчающие процесс разработки, что делает среду Visual Studio одной из самых простых и понятных в освоении и 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 функционирование алгоритма сортировки методом «пузырька». Итоговая программа должна непосредственно воплощать данный алгоритм, а также осуществлять получение исходных данных из файла и сохранение полученных результатов в файл.

# 1.1 Достоинства алгоритма

Алгоритм сортировки "пузырьком" выделяется своей простотой и доступностью для понимания, что делает его популярным среди начинающих программистов. Благодаря своей элементарной структуре, он может быть без труда реализован на различных языках программирования.

Стоит отметить, что метод "пузырька" служит фундаментом для создания более сложных и эффективных алгоритмов сортировки, демонстрируя его значимость в области алгоритмизации. Простота реализации и концептуальная ясность делают его отличной отправной точкой для изучения принципов сортировки данных.

# 1.2 Недостатки алгоритма

В силу своей простоты, алгоритм сортировки пузырьком в основном рассматривается в образовательных целях и редко используется в реальных приложениях, особенно при обработке значительных объемов данных. Это связано с тем, что алгоритм требует множественных итераций и сравнений на каждом этапе, что приводит к существенным затратам времени.

Время, необходимое для выполнения сортировки пузырьком, возрастает квадратично по отношению к количеству элементов (N^2), что делает данный метод одним из наименее производительных алгоритмов сортировки. Таким образом, для больших наборов данных существуют гораздо более эффективные альтернативы.

# 1.3 Типичные сценарии использования

Этот алгоритм сортировки подходит для образовательных целей, особенно для начинающих программистов, а также эффективен для небольших объемов данных. Он может быть использован для оптимизации поиска элементов или для систематизации информации.

# 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 настоящего исследования студент Любченко В.К. занимался реализацией практической задачи, а именно алгоритма «Сортировка пузырьком». В ходе выполнения поставленной задачи, студент осуществил процедуру считывания информации из предварительно заданного файла и разработал программный код, реализующий алгоритм сортировки.

#### Алгоритм сортировки пузырьком (Рис 1.)

def bubble\_sort(self):

# Проверка на возможность сортировки

if self.repeat\_sort:

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

return

if not self.arr:

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

return

# Замер времени выполнения

start = time.time()

# Основной алгоритм сортировки

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]:

# Обмен элементов местами

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

swapped = True

# Оптимизация: выход при отсутствии обменов

if not swapped:

break

# Фиксация времени выполнения

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:.6f} сек.")

**Сохранение массива в файл (Рис 2.)**

def save\_array(self):

# Проверка наличия данных

if not self.arr:

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

return

# Диалог выбора файла

filename = filedialog.asksaveasfilename(

defaultextension=".txt",

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

)

if not filename:

return

# Запись данных в файл

try:

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

f.write(f"{len(self.arr)}\n") # Запись размера массива

f.write(" ".join(map(str, self.arr))) # Запись элементов через пробел

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

except Exception as e:

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

**Формат файла:**

[количество элементов]

[элемент1] [элемент2] ... [элементN]

# Схема программы

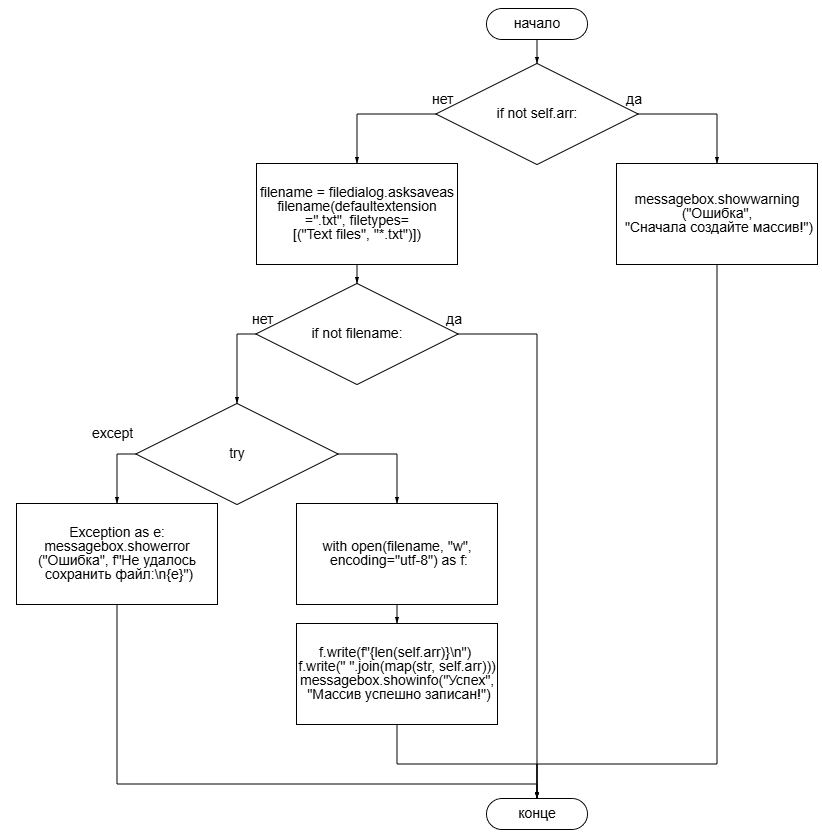


Рисунок 1 - Схема алгоритма save\_array

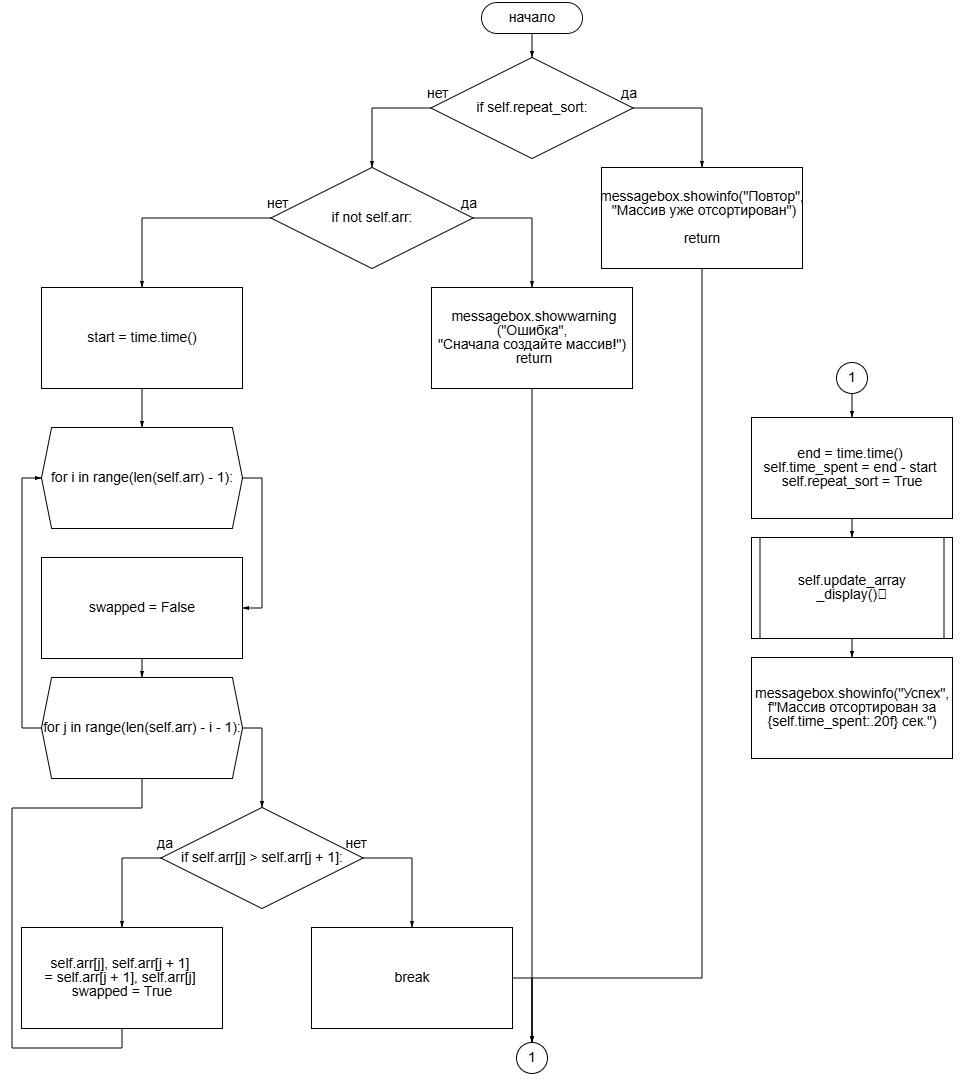


Рисунок 2 - Схема алгоритма bubble\_sort

# 5. Отладка

В качестве инструментария для разработки использовалась среда Microsoft Visual Studio, интегрирующая в себе полный набор средств, необходимых для создания и отладки программных модулей и приложений.

Процесс отладки программного кода включал в себя применение точек останова, обеспечивающих приостановку выполнения программы в заданных местах, а также пошаговое выполнение кода, позволяющее детально отслеживать ход выполнения операций. Дополнительно проводился анализ значений глобальных и локальных переменных для выявления возможных ошибок в логике работы программы.

Тестирование программного продукта осуществлялось на протяжении всего цикла разработки, начиная с этапа написания отдельных модулей и заканчивая финальной стадией разработки. После завершения кодирования, независимым специалистом по тестированию проводилась проверка работоспособности программы, в ходе которой были обнаружены и устранены ошибки и недочеты.

# 6. Совместная разработка

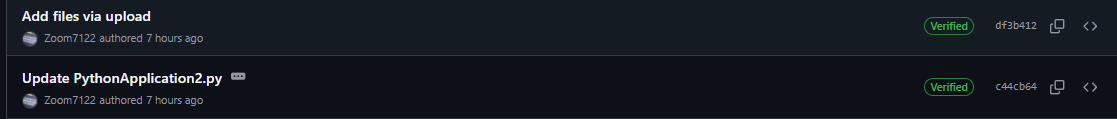
В рамках выполнения данного практического задания, наша команда осуществляла коллаборативную деятельность, используя платформу GitHub. Изначальная версия алгоритма была разработана участником команды, Любченко В.К. По завершении процесса написания, программный код был загружен в удаленный репозиторий GitHub, в основную ветку под названием "main".

Рисунок 7 – Загрузка программы на GitHub

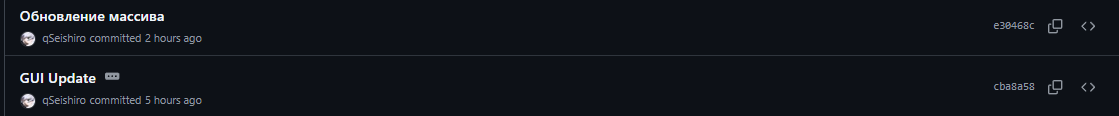
Впоследствии, участник Кукушкин А.А. осуществил загрузку указанного программного обеспечения на свой персональный компьютер, используя команду git clone <ссылка>. Им было внесено дополнение, включающее разработку пользовательского интерфейса, а также произведено обновление массива данных. Кроме того, Кукушкин А.А. выполнил работы по оптимизации кода. 

Рисунок 8 – Загрузка доработанного кода на GitHub

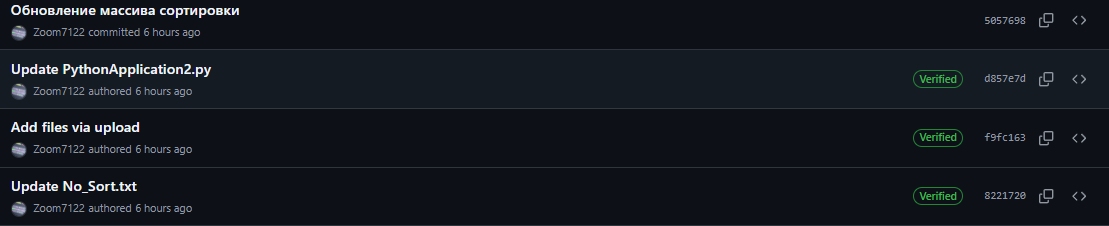
Впоследствии, Любченко В.К. внес коррективы в незначительные ошибки в программном коде и интегрировал процедуру верификации.

Рисунок 9 – Исправление мелких недочетов

# Впоследствии, третий член команды, Комиссаров А.Г., осуществил проверку работоспособности программного обеспечения и разработал соответствующие блок-схемы. Все изображения экрана были помещены в отдельные директории в главной ветви проекта.

# Заключение

В процессе выполнения данной учебно-практической работы были изучены ключевые концепции и принципы работы алгоритма пузырьковой сортировки, а также разработан механизм взаимодействия с файлами. Наглядно продемонстрирована функциональность алгоритма сортировки методом "пузырька". Итоговая программа не только реализует сам алгоритм, но и обеспечивает чтение исходных данных из файла, а также запись полученных результатов в файл.

В ходе тестирования была проведена отладка программного кода, подтвердившая корректность работы алгоритма. Предусмотрены ограничения на формат входных числовых данных и обработка ситуаций, связанных с ошибками чтения файлов. Результаты тестирования указывают на целесообразность использования данного алгоритма для сортировки массивов небольшого размера. При увеличении количества сортируемых элементов наблюдается экспоненциальный рост времени выполнения программы.

Сортировка "пузырьком" представляет собой достаточно простой и понятный алгоритм, эффективный при работе с небольшими последовательностями чисел, частично упорядоченными данными, а также в образовательных целях. Несмотря на легкость реализации, высокая вычислительная сложность и значительные временные затраты, требуемые для выполнения сортировки, существенно ограничивают область практического 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

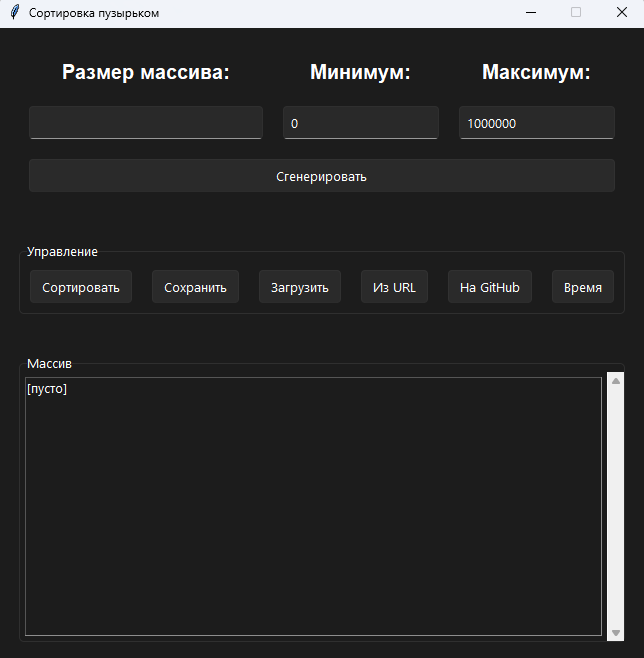


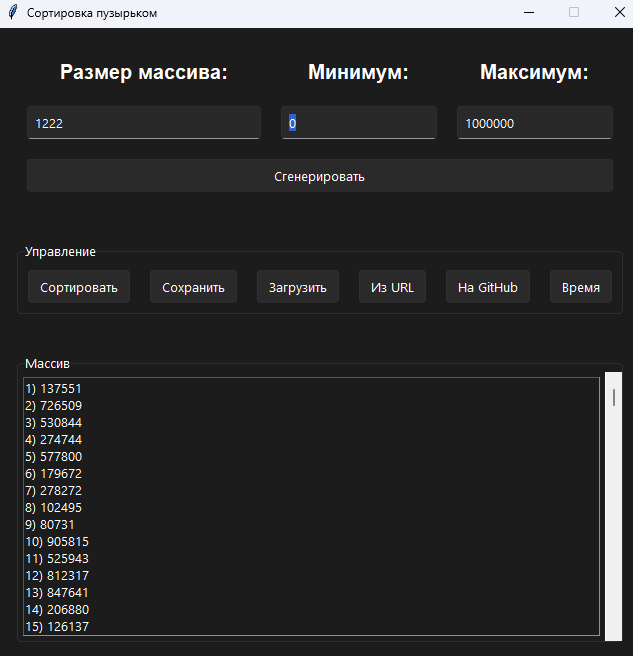
Рисунок А.1 - главный экран

Рисунок А.2 - Созданный массив

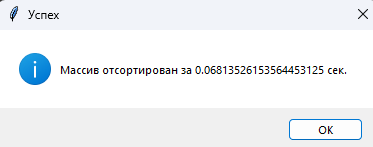


Рисунок А.3 - Успех сортировки

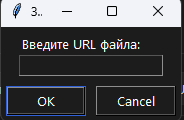


Рисунок А.4 - ввод массива по URL

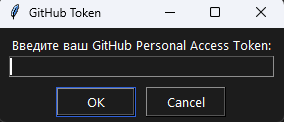


Рисунок А.5 - ввод массива из githab

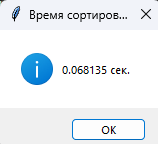


Рисунок А.6 - Кнопка “Время”

# Приложение Б. Листинг

**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}")

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()