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)

\_\_\_\_ Агапов Иван Андреевич

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 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)

                                            Агапов Иван Андреевич

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 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 | 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)

Агапов Иван Андреевич

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»

Агапов И.А. выполнял практическое задание «Шейкерная сортировка». На первоначальном этапе были изучен и проанализирован алгоритм шейкерной сортировки, был выбран метод решения и язык программирования C, на котором была написана программа шейкерной сортировки массива. Также, осуществил подсчет времени выполнения сортировки. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Агапов И.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)

Агапов Иван Андреевич

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 процессе выполнения практики Агапов И.А. решал следующие задачи: изучение алгоритма шейкерной сортировки, анализ работы алгоритма, сравнение существующих методов сортировки, произведение подсчета времени выполнения сортировки, тестирование и отладка программы.

За период выполнения практики были освоены основные понятия и технологии шейкерной сортировки. Во время выполнения работы Агапов И.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 Агапов И.А. заслуживает оценки «\_\_\_\_\_\_».

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

**Содержание**

[**Введение** 2](#_Toc202722178)

[**1** **Постановка задачи** 3](#_Toc202722179)

[**1.1** **Достоинства шейкерной сортировки** 3](#_Toc202722180)

[**1.2** **Недостатки шейкерной сортировки** 3](#_Toc202722181)

[**1.3** **Типичные сценарии применения данного алгоритма** 3](#_Toc202722182)

[**2** **Выбор решения** 4](#_Toc202722183)

[**3** **Описание программы** 5](#_Toc202722184)

[**4** **Схемы программы** 8](#_Toc202722185)

[**4.1** **Блок-схема программы** 8](#_Toc202722186)

[**4.2** **Блок-схема алгоритма** 9](#_Toc202722187)

[**5** **Тестирование программы** 10](#_Toc202722188)

[**5.1** **Тестирование на разных наборах данных** 10](#_Toc202722189)

[**5.2 Анализ полученных результатов тестирования** 10](#_Toc202722190)

[**6** **Отладка** 12](#_Toc202722191)

[**7** **Совместная разработка** 13](#_Toc202722192)

[**Заключение** 16](#_Toc202722193)

[**Список используемой литературы** 17](#_Toc202722194)

[**Приложение А. Результаты тестирования программы** 18](#_Toc202722195)

[**Приложение Б. Листинг программы** 20](#_Toc202722196)

# **Введение**

Сортировка данных в современном мире компьютерных технологий представляет собой один из ключевых процессов обработки информации. Она находит применение во множестве профессиональных сфер — от анализа больших данных до оптимизации работы баз данных.

Алгоритмы сортировки образуют отдельный класс методов, которые не только упорядочивают элементы, но и обеспечивают основу для эффективного поиска и дальнейшей обработки.

Существует множество алгоритмов сортировки, каждый из которых обладает уникальными характеристиками. Например, сортировка выбором проста в реализации, но неэффективна на больших объемах данных, тогда как быстрая сортировка, несмотря на свою сложность, обеспечивает высокую скорость работы. Сортировка вставками, хоть и уступает по производительности более продвинутым методам, остается стабильной и часто используется в комбинации с другими алгоритмами для создания гибридных решений. Это разнообразие позволяет выбирать оптимальный подход в зависимости от конкретной задачи.

Шейкерная сортировка, или сортировка перемешиванием, представляет собой усовершенствованную версию пузырькового алгоритма. Её ключевая особенность — двунаправленное движение по массиву: сначала слева направо, затем справа налево. Такой подход позволяет эффективно "проталкивать" крупные элементы к концу массива и одновременно "вытягивать" мелкие элементы к началу, что ускоряет процесс упорядочивания.

# **Постановка задачи**

Поставленная задача: необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить шейкерную сортировку над данными, находящимися в массиве, записать отсортированные данные в другой файл, посчитать время выполнения сортировки.

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

Оформить отчет по проведенной практике.

## **Достоинства шейкерной сортировки**

* алгоритм эффективнее классической пузырьковой сортировки за счет двунаправленного прохода по данным
* простая и понятная реализация алгоритма
* не требует дополнительной памяти для выполнения сортировки

## **Недостатки шейкерной сортировки**

* низкая эффективность на больших массивах
* медленная работа на случайных данных

## **Типичные сценарии применения данного алгоритма**

* спортивные рейтинги (сортировка спортсменов по личным рекордам, времени в предварительных заездах)
* музыка (сортировка песен по дате добавления, продолжительности песен)
* торговля (сортировка товаров по цене, сроку годности, весу)

# **Выбор решения**

Для написание данной программы будет использован язык программирования Си. Этот язык является распространенным языком программирования. Этот язык по праву считается одним из фундаментальных в компьютерной науке, занимая особое положение между низкоуровневым ассемблером и высокоуровневыми языками программирования. Си сочетает в себе мощь и контроль над аппаратными ресурсами с относительной простотой и выразительностью кода.

Для разработки проекта была выбрана интегрированная среда Microsoft Visual Studio — мощный инструмент, предоставляющий все необходимые средства для создания как консольных приложений, так и программ с графическим интерфейсом.

# **Описание программы**

При запуске программы выводится меню в котором можно выбрать способ заполнения массива (случайная генерация или ручной ввод). После выбора способа выводится меню из семи пунктов:

1) изменить размер массива;

2) просмотреть исходный массив;

3) просмотреть отсортированный массив;

4) очистить файлы;

5) изменить диапазон массива;

6) показать время последней сортировки;

7) выйти из программы.

Пользователю требуется выбрать тот пункт, который ему нужен. При выборе пункта способа заполнения массива выводится сообщение, в котором пользователю необходимо ввести размер массива и указать диапазон значений.

printf(" Введите размер массива: ");

scanf("%d", &n);

После того, как был введен размер массива, указывается диапазон значений.

printf(" Введите минимальное значение элемента: ");

scanf("%d", &min\_val);

printf(" Введите максимальное значение элемента: ");

scanf("%d", &max\_val);

Далее происходит генерация массива.

srand(time(0));

int range = max\_val - min\_val + 1;

for (int i = 0; i < n; i++) {

arr[i] = rand() % range + min\_val;

}

Эти числа записываются в input.txt.

Далее над этими данными выполняется шейкерная сортировка, при которой массив перебирается не только слева направо, но и справа налево. Алгоритм начинает с первого элемента и сравнивает его со следующим. Если текущий элемент больше следующего, то они меняются местами. Так продолжается до конца массива, пока наибольший элемент не встанет в конец. Далее алгоритм идет в обратном направлении и сравнивает элементы, перемещая самый маленький элемент в начало.

Алгоритм сортировки:

int left = 0;

int right = n - 1;

int swapped = 1;

while (left < right&& swapped) {

swapped = 0;

for (int i = left; i < right; i++) {

if (arr[i] > arr[i + 1]) {

swap(&arr[i], &arr[i + 1]);

swapped = 1;

}

}

right--;

for (int i = right; i > left; i--) {

if (arr[i] < arr[i - 1]) {

swap(&arr[i], &arr[i - 1]);

swapped = 1;

}

}

left++;

if (!swapped) break;

}

После этого отсортированный массив записывается в файл sorted.txt.

Программа также осуществляет подсчет времени, которое заняла сортировка.

При выборе пункта меню под цифрой 7 программа завершает выполнение.

Подробный алгоритм программы и функции сортировки представлен в разделе 4 на рисунках 1, 2.

Листинг программы приведен в приложении Б.

# **Схемы программы**

## **Блок-схема программы**

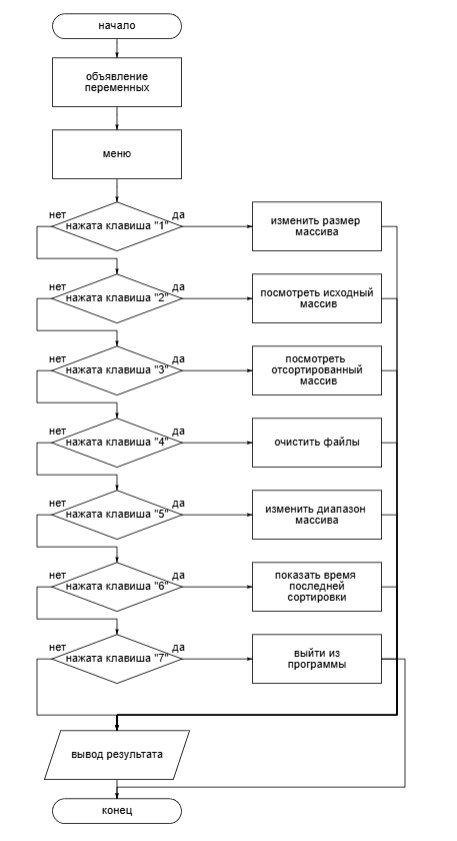


Рисунок 1 - Блок-схема программы

## **Блок-схема алгоритма**

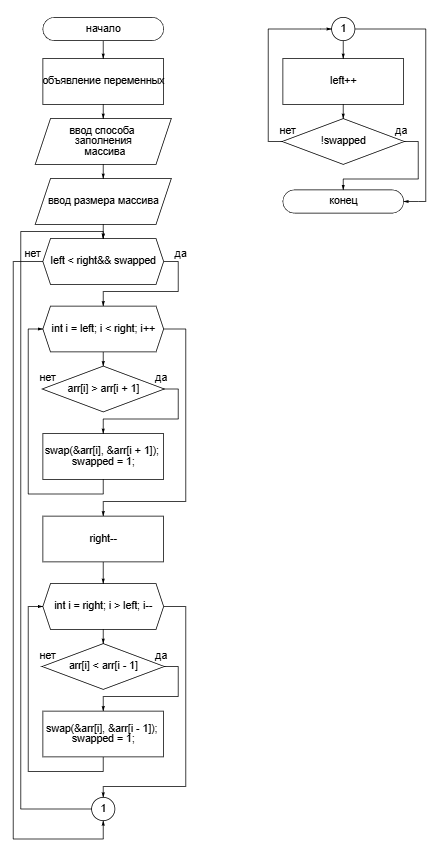


Рисунок 2 - Блок-схема алгоритма

# **Тестирование программы**

## **Тестирование на разных наборах данных**

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в приложении А на рисунках А.1 – А.5.

Таблица 1 - Тестовый набор данных

|  |  |  |
| --- | --- | --- |
|  | Размер массива | Время выполнения сортировки в секундах |
| 1 | 10000 | 0.142 |
| 2 | 20000 | 0.655 |
| 3 | 30000 | 1.607 |
| 4 | 40000 | 2.951 |
| 5 | 50000 | 4.802 |

## **5.2 Анализ полученных результатов тестирования**

Тестирование показало, что с увеличением количества элементов пропорционально увеличивается время работы программы. График зависимости времени выполнения сортировки от количества элементов в наборе приведен на рисунке 3.

Рисунок 3 - Результаты тестирования

# **Отладка**

В качестве среды разработки была выбрана программа Microsoft Visual Studio, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ.

Для отладки программы использовались точки остановки и пошаговое выполнение кода программы, анализ содержимого локальных переменных.

Точки останова – это прерывание выполнения программы, при котором выполняется вызов отладчика. Отладчик является инструментом для поиска устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Команда шаг с заходом (step into) выполняет следующую инструкцию в обычном пути выполнения программы, а затем приостанавливает выполнение программы, чтобы мы могли проверить состояние программы с помощью отладчика. Если выполняемый оператор содержит вызов функции, шаг с заходом заставляет программу перескакивать в начало вызываемой функции, где она приостанавливается.

# **Совместная разработка**

Для удобства совместной разработки был использован GitHub.

Определили задачи проекта.

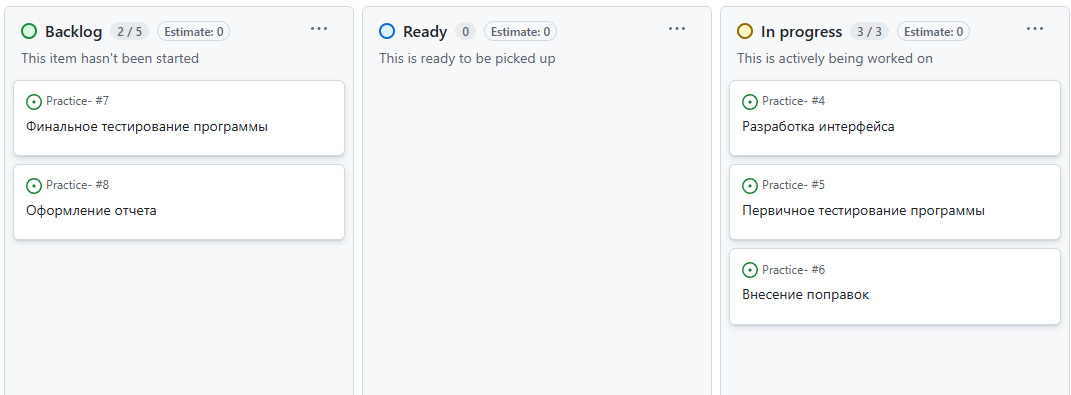


Рисунок 4 - Определение задач проекта

Разделили роли, назначили исполнителей задачам.

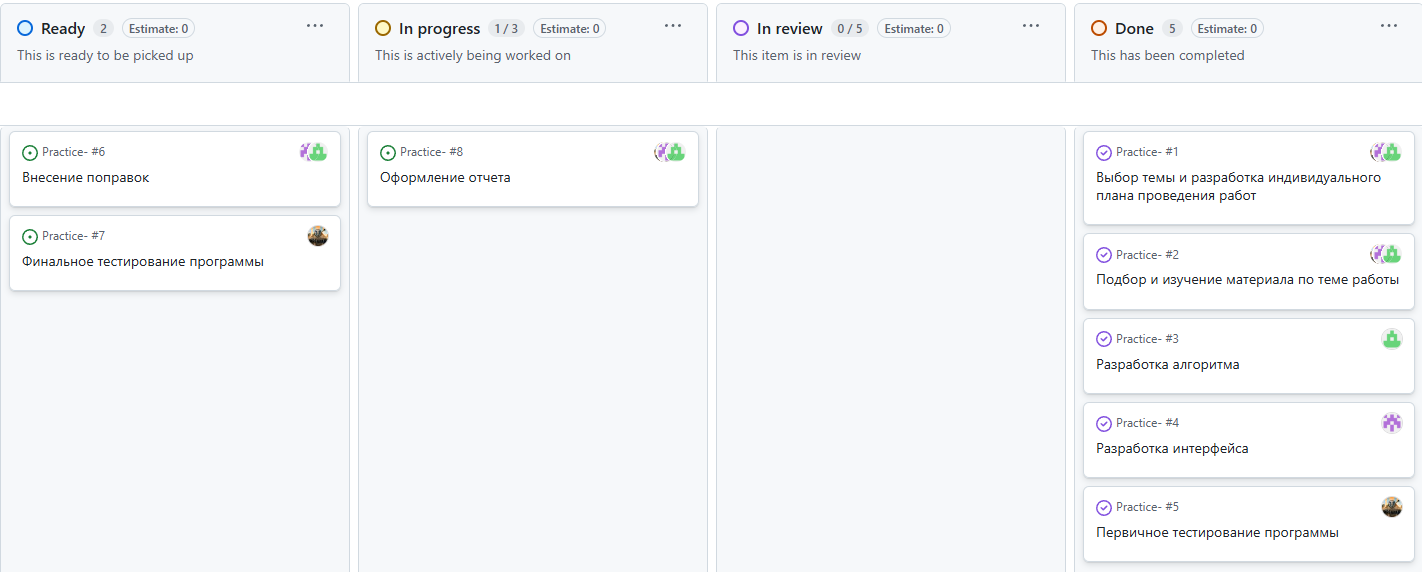


Рисунок 5 - Распределение задач проекта

Корректировали статус задач по мере выполнения.

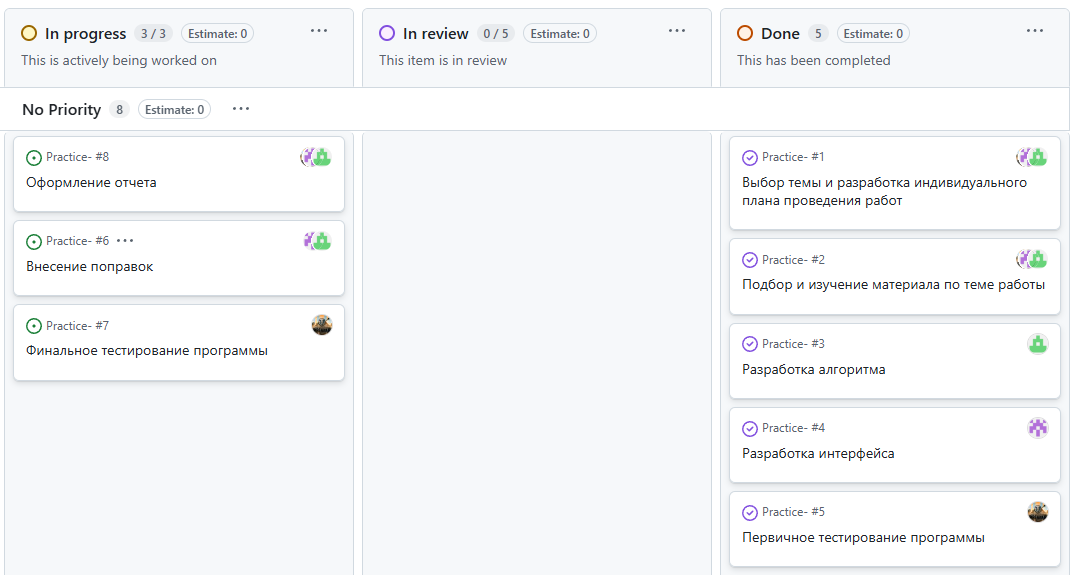


Рисунок 6 - Корректирование задач

В GitHub были выданы задачи и сроки в которые наша группа должна была выполнить ряд заданий.

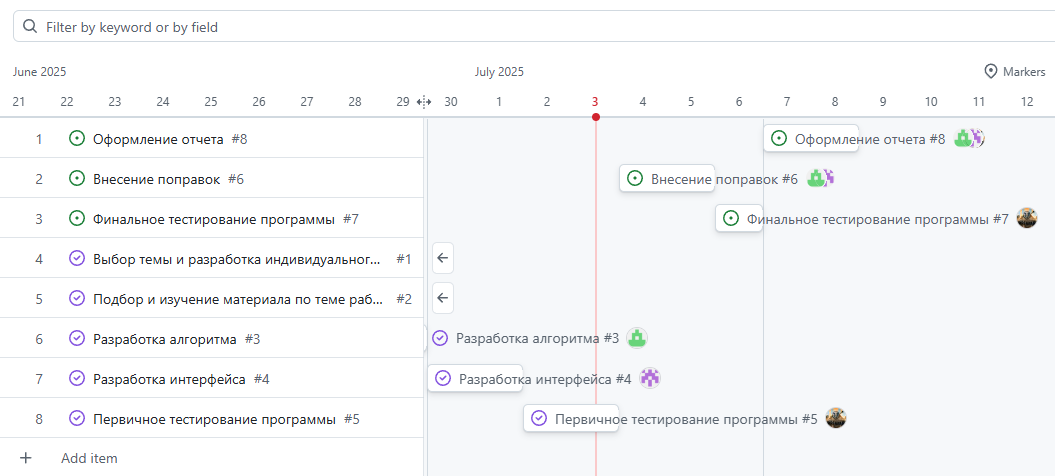


Рисунок 7 - дорожная карта действий группы

Мною было проведено тестирование программы и загрузка отчета в удаленный репозиторий GitHub, на ветку main.

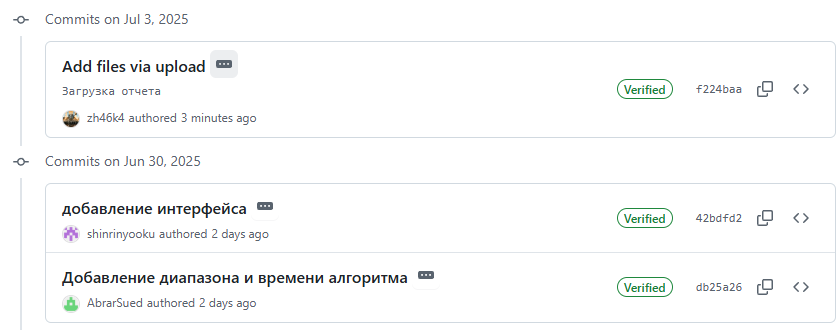


Рисунок 8 - Созданные коммиты

Ссылка на удаленный репозиторий:

<https://github.com/AbrarSued/Practice-.git>

# **Заключение**

При выполнении данной работы были получены навыки совместной работы с помощью сервиса GitHub. Был изучен шейкерный алгоритм.

Мною был написан алгоритм, считающий время работы программы в секундах. Было выполнено тестирование программы на разных наборах данных и отладка данной программы.

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

# **Список используемой литературы**

1. ГОСТ 19.701 – 90 схемы алгоритмов, программ, данных и систем.
2. Керниган, Брайан У., Ритчи, Денис М. Язык программирования С, 2-е издание.: Пер. с англ. – М., 2009.
3. Сортировка перемешиванием [Электронный ресурс] – URL: <https://ru.wikipedia.org> (дата обращения 29.07.2025 г)

# **Приложение А. Результаты тестирования программы**

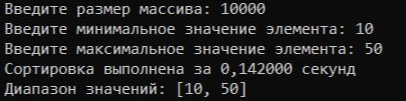


Рисунок А.1

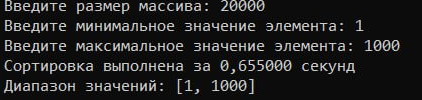


Рисунок А.2

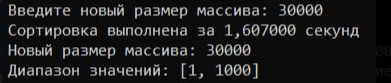


Рисунок А.3

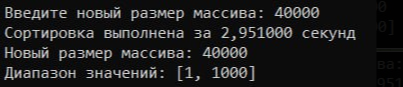


Рисунок А.4

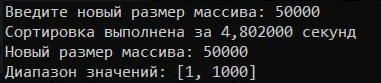


Рисунок А.5

# **Приложение Б. Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <string.h>

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void shakerSort(int arr[], int n) {

int left = 0;

int right = n - 1;

int swapped = 1;

while (left < right&& swapped) {

swapped = 0;

for (int i = left; i < right; i++) {

if (arr[i] > arr[i + 1]) {

swap(&arr[i], &arr[i + 1]);

swapped = 1;

}

}

right--;

for (int i = right; i > left; i--) {

if (arr[i] < arr[i - 1]) {

swap(&arr[i], &arr[i - 1]);

swapped = 1;

}

}

left++;

if (!swapped) break;

}

}

void printCentered(const char\* text, int width) {

int len = strlen(text);

int padding = (width - len) / 2;

printf("+");

for (int i = 0; i < width; i++) printf("-");

printf("+\n");

printf("|");

for (int i = 0; i < padding; i++) printf(" ");

printf("%s", text);

for (int i = 0; i < width - len - padding; i++) printf(" ");

printf("|\n");

printf("+");

for (int i = 0; i < width; i++) printf("-");

printf("+\n");

}

void printArrayFromFile(const char\* filename, const char\* title, int width) {

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Ошибка открытия файла %s\n", filename);

return;

}

printCentered(title, width - 2);

int num, count = 0;

while (fscanf(file, "%d", &num) == 1) {

printf(" %5d", num);

count++;

if (count % 15 == 0) {

printf("\n");

}

}

if (count % 15 != 0) {

printf("\n");

}

fclose(file);

}

void saveToFile(int arr[], int n, const char\* filename) {

FILE\* file = fopen(filename, "w");

if (file == NULL) {

printf("Ошибка открытия файла %s\n", filename);

return;

}

for (int i = 0; i < n; i++) {

fprintf(file, "%d ", arr[i]);

if ((i + 1) % 15 == 0) {

fprintf(file, "\n");

}

}

fclose(file);

}

int createArray(int\*\* arr\_ptr, int\* n\_ptr, int\* min\_val\_ptr, int\* max\_val\_ptr, int\* generation\_type) {

int choice;

printf("\n Выберите способ заполнения массива:\n");

printf(" 1 - Случайная генерация\n");

printf(" 2 - Ручной ввод\n");

printf(" Ваш выбор: ");

scanf("%d", &choice);

if (choice != 1 && choice != 2) {

printf("\nОшибка: неверный выбор!\n");

return 0;

}

\*generation\_type = choice;

int n;

printf(" Введите размер массива: ");

scanf("%d", &n);

if (n <= 0) {

printf("\nОшибка: размер массива должен быть положительным!\n");

return 0;

}

int\* arr = (int\*)malloc(n \* sizeof(int));

if (arr == NULL) {

printf("\nОШИБКА ВЫДЕЛЕНИЯ ПАМЯТИ!\n");

return 0;

}

if (choice == 1) {

int min\_val, max\_val;

printf(" Введите минимальное значение элемента: ");

scanf("%d", &min\_val);

printf(" Введите максимальное значение элемента: ");

scanf("%d", &max\_val);

if (min\_val > max\_val) {

printf("\nОшибка: минимальное значение не может быть больше максимального!\n");

free(arr);

return 0;

}

srand(time(0));

int range = max\_val - min\_val + 1;

for (int i = 0; i < n; i++) {

arr[i] = rand() % range + min\_val;

}

\*min\_val\_ptr = min\_val;

\*max\_val\_ptr = max\_val;

}

else if (choice == 2) {

printf("\n Введите элементы массива:\n");

for (int i = 0; i < n; i++) {

printf(" Элемент [%d]: ", i + 1);

scanf("%d", &arr[i]);

}

int min\_val = arr[0];

int max\_val = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] < min\_val) min\_val = arr[i];

if (arr[i] > max\_val) max\_val = arr[i];

}

\*min\_val\_ptr = min\_val;

\*max\_val\_ptr = max\_val;

}

\*arr\_ptr = arr;

\*n\_ptr = n;

saveToFile(arr, n, "input.txt");

return 1;

}

int resizeArray(int\*\* arr\_ptr, int\* n\_ptr, int generation\_type, int min\_val, int max\_val) {

int new\_n;

printf(" Введите новый размер массива: ");

scanf("%d", &new\_n);

if (new\_n <= 0) {

printf("\nОшибка: размер массива должен быть положительным!\n");

return 0;

}

int\* new\_arr = (int\*)malloc(new\_n \* sizeof(int));

if (new\_arr == NULL) {

printf("\nОШИБКА ВЫДЕЛЕНИЯ ПАМЯТИ!\n");

return 0;

}

if (generation\_type == 1) {

srand(time(0));

int range = max\_val - min\_val + 1;

for (int i = 0; i < new\_n; i++) {

new\_arr[i] = rand() % range + min\_val;

}

}

else if (generation\_type == 2) {

printf("\n Введите новые элементы массива:\n");

for (int i = 0; i < new\_n; i++) {

printf(" Элемент [%d]: ", i + 1);

scanf("%d", &new\_arr[i]);

}

}

if (\*arr\_ptr) {

free(\*arr\_ptr);

}

\*arr\_ptr = new\_arr;

\*n\_ptr = new\_n;

saveToFile(new\_arr, new\_n, "input.txt");

return 1;

}

void changeRange(int\* arr, int n, int\* min\_val\_ptr, int\* max\_val\_ptr) {

int min\_val, max\_val;

printf(" Введите новый минимальный элемент: ");

scanf("%d", &min\_val);

printf(" Введите новый максимальный элемент: ");

scanf("%d", &max\_val);

if (min\_val > max\_val) {

printf("\nОшибка: минимальное значение не может быть больше максимального!\n");

return;

}

srand(time(0));

int range = max\_val - min\_val + 1;

for (int i = 0; i < n; i++) {

arr[i] = rand() % range + min\_val;

}

\*min\_val\_ptr = min\_val;

\*max\_val\_ptr = max\_val;

saveToFile(arr, n, "input.txt");

printf(" Диапазон массива изменен\n");

}

void calculateMinMax(int arr[], int n, int\* min\_val, int\* max\_val) {

if (n <= 0) return;

\*min\_val = arr[0];

\*max\_val = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] < \*min\_val) \*min\_val = arr[i];

if (arr[i] > \*max\_val) \*max\_val = arr[i];

}

}

int main() {

setlocale(LC\_ALL, "");

int n = 0;

int\* arr = NULL;

int min\_val = 0, max\_val = 0;

int generation\_type = 0;

const int consoleWidth = 90;

clock\_t start, end;

double time\_used = 0;

printCentered("ШЕЙКЕРНАЯ СОРТИРОВКА", consoleWidth - 2);

if (!createArray(&arr, &n, &min\_val, &max\_val, &generation\_type)) {

return 1;

}

start = clock();

shakerSort(arr, n);

end = clock();

time\_used = ((double)(end - start)) / CLOCKS\_PER\_SEC;

saveToFile(arr, n, "sorted.txt");

printf(" Сортировка выполнена за %.6f секунд\n", time\_used);

printf(" Диапазон значений: [%d, %d]\n", min\_val, max\_val);

int menu\_choice;

while (1) {

printf("\n");

printCentered("ГЛАВНОЕ МЕНЮ", consoleWidth - 2);

printf(" Текущий размер массива: %d\n", n);

printf(" Текущий способ заполнения: %s\n", generation\_type == 1 ? "Случайный" : "Ручной");

if (generation\_type == 1) {

printf(" Текущий диапазон значений: [%d, %d]\n", min\_val, max\_val);

}

printf(" 1 - Изменить размер массива\n");

printf(" 2 - Просмотреть исходный массив\n");

printf(" 3 - Просмотреть отсортированный массив\n");

printf(" 4 - Очистить файлы\n");

printf(" 5 - Изменить диапазон массива\n");

printf(" 6 - Показать время последней сортировки\n");

printf(" 7 - Выйти из программы\n");

printf(" Ваш выбор: ");

scanf("%d", &menu\_choice);

switch (menu\_choice) {

case 1:

if (!resizeArray(&arr, &n, generation\_type, min\_val, max\_val)) {

break;

}

calculateMinMax(arr, n, &min\_val, &max\_val);

start = clock();

shakerSort(arr, n);

end = clock();

time\_used = ((double)(end - start)) / CLOCKS\_PER\_SEC;

saveToFile(arr, n, "sorted.txt");

printf(" Сортировка выполнена за %.6f секунд\n", time\_used);

printf(" Новый размер массива: %d\n", n);

if (generation\_type == 1) {

printf(" Диапазон значений: [%d, %d]\n", min\_val, max\_val);

}

break;

case 2:

printArrayFromFile("input.txt", "ИСХОДНЫЙ МАССИВ", consoleWidth);

break;

case 3:

printArrayFromFile("sorted.txt", "ОТСОРТИРОВАННЫЙ МАССИВ", consoleWidth);

break;

case 4:

remove("input.txt");

remove("sorted.txt");

printf(" Файлы input.txt и sorted.txt успешно удалены\n");

break;

case 5:

if (generation\_type != 1) {

printf(" Изменение диапазона доступно только для случайно сгенерированных массивов!\n");

break;

}

changeRange(arr, n, &min\_val, &max\_val);

start = clock();

shakerSort(arr, n);

end = clock();

time\_used = ((double)(end - start)) / CLOCKS\_PER\_SEC;

saveToFile(arr, n, "sorted.txt");

printf(" Сортировка выполнена за %.6f секунд\n", time\_used);

printf(" Новый диапазон: [%d, %d]\n", min\_val, max\_val);

break;

case 6:

printf(" Время последней сортировки: %.6f секунд\n", time\_used);

break;

case 7:

if (arr) free(arr);

return 0;

default:

printf(" Неверный выбор! Попробуйте снова.\n");

}

}

}