**Содержание**

[**Введение** 2](#_Toc202395156)

[**1** **Постановка задачи** 3](#_Toc202395157)

[**1.1** **Достоинства шейкерной сортировки** 3](#_Toc202395158)

[**1.2** **Недостатки шейкерной сортировки** 3](#_Toc202395159)

[**1.3** **Типичные сценарии применения данного алгоритма** 3](#_Toc202395160)

[**2** **Выбор решения** 4](#_Toc202395161)

[**3** **Описание программы** 5](#_Toc202395162)

[**4** **Схемы программы** 8](#_Toc202395163)

[**4.1** **Блок-схема программы** 8](#_Toc202395164)

[**4.2** **Блок-схема алгоритма** 9](#_Toc202395165)

[**5** **Тестирование программы** 10](#_Toc202395166)

[**6** **Отладка** 11](#_Toc202395167)

[**7** **Совместная разработка** 12](#_Toc202395168)

# **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 A.

# **Схемы программы**

## **Блок-схема программы**

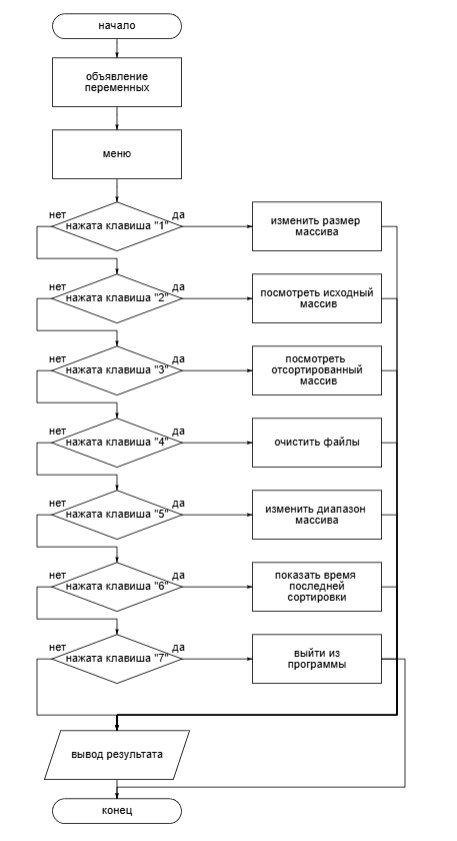


Рисунок 1 - Блок-схема программы

## **Блок-схема алгоритма**

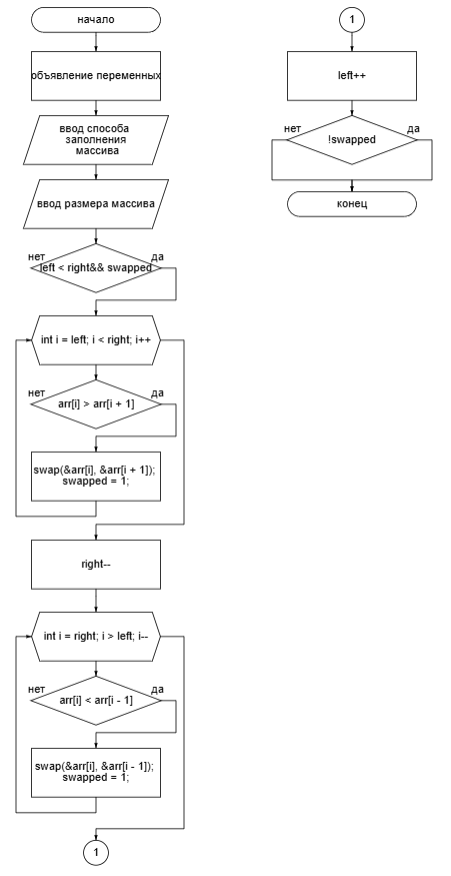


Рисунок 2 - Блок-схема алгоритма

# **Тестирование программы**

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.

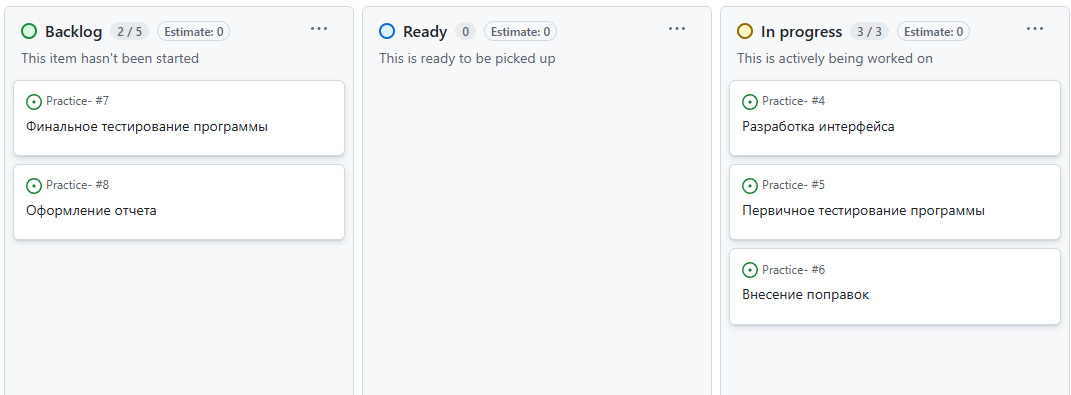


Рисунок 4 - Определение задач проекта

Разделили роли, назначили исполнителей задачам.

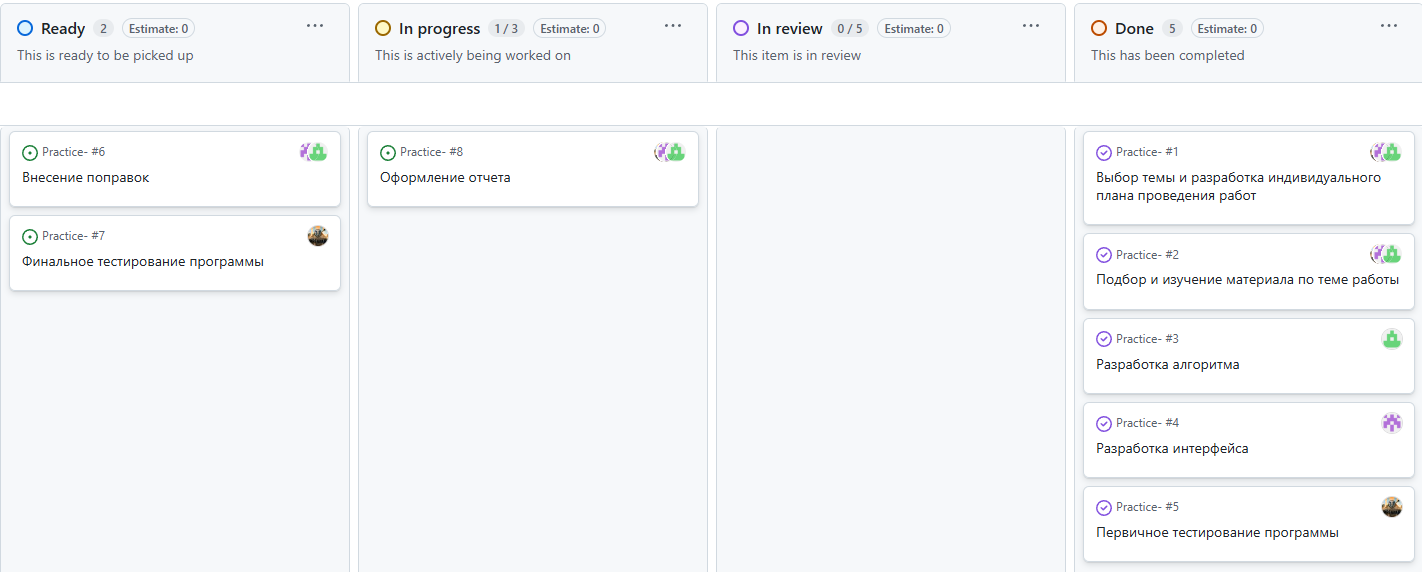


Рисунок 5 - Распределение задач проекта

Корректировали статус задач по мере выполнения.

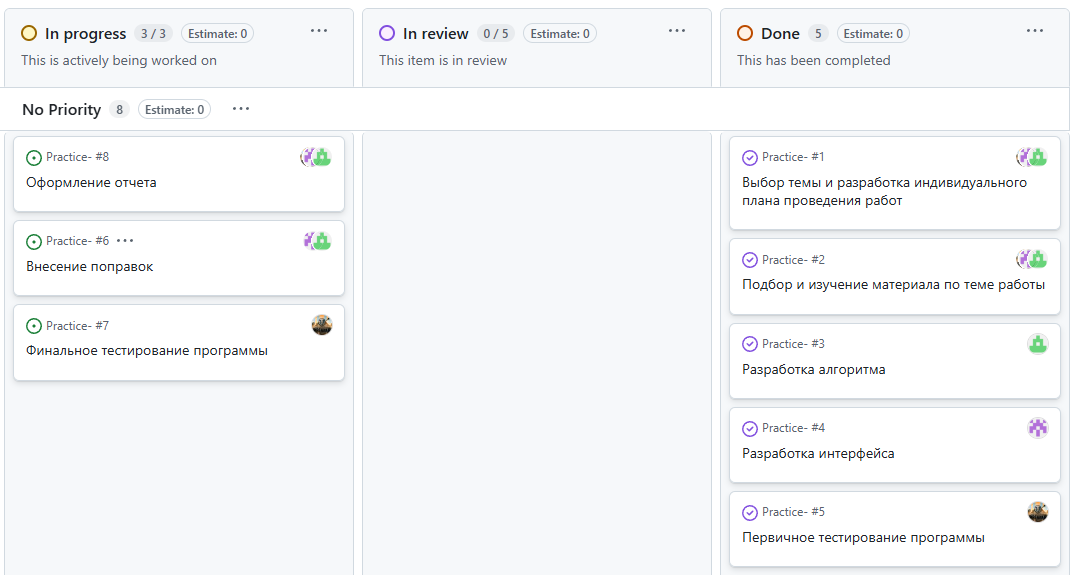


Рисунок 6 - Корректирование задач

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

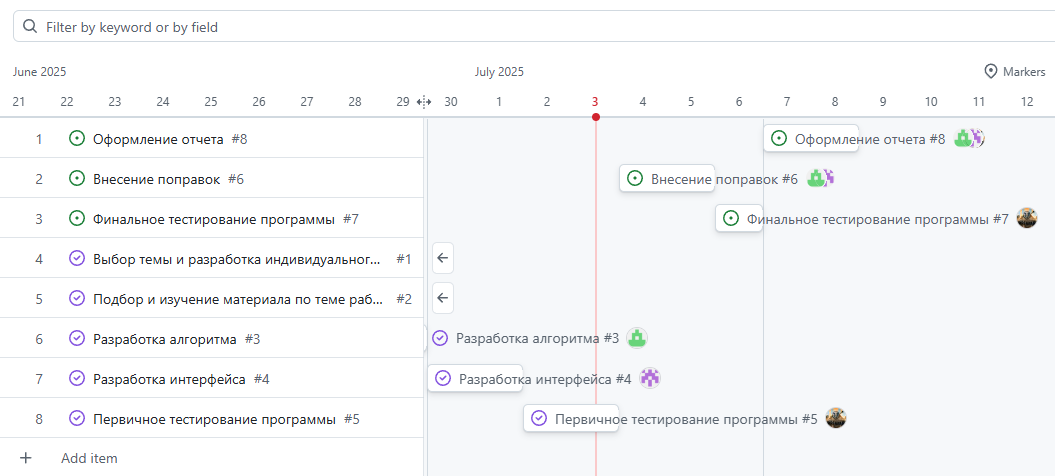


Рисунок 7 - дорожная карта действий группы