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)

                                               Бегичев Денис Дмитриевич

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)

                                            Бегичев Денис Дмитриевич

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)

                                            Бегичев Денис Дмитриевич

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»

Бегичев Д.Д.. выполнял практическое задание «Сортировка пузырьком». На первом этапе была изучен и проанализирован алгоритм сортировки пузырьком и сбор метрик. Был выбран язык программирования C, на котором была написана программа с удобным консольным меню для взаимодействия с пользователем. В процессе работы была реализован функция сортировки массива и сбор метрик (секунды и количество перестановок). Программа была тщательно протестирована и отлажена на различных входных данных. По итогам работы был оформлен подробный отчет.

Бакалавр Бегичев Д.Д.. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                            Бегичев Денис Дмитриевич

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 Бегичев Д.Д. решал следующие задачи: создание алгоритма пузырьковой сортировки, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии пузырьковой сортировки, реализован метод сбора метрик. Во время выполнения работы Бегичев Д.Д. 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 Бегичев Д.Д. заслуживает оценки «\_\_\_\_\_\_».

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

# **Содержание**

[Введение 3](#_Toc202431633)

[1 Постановка задачи 4](#_Toc202431634)

[1.1 Достоинства алгоритма сортировки пузырьком 4](#_Toc202431635)

[1.2 Недостатки алгоритма сортировки пузырьком 4](#_Toc202431636)

[1.3 Типичные сценария применения данного алгоритма 5](#_Toc202431637)

[2 Выбор решения 6](#_Toc202431638)

[3 Описание программы 7](#_Toc202431639)

[4 Схемы программы 11](#_Toc202431640)

[4.1 Блок-схема программы 11](#_Toc202431641)

[4.2 Блок-схема алгоритма 12](#_Toc202431642)

[5 Тестирование программы 13](#_Toc202431643)

[6 Отладка 14](#_Toc202431644)

[7 Совместная разработка 15](#_Toc202431645)

[Заключение 18](#_Toc202431646)

[Список используемой литературы 19](#_Toc202431647)

[Приложение А. Листинг программы 20](#_Toc202431648)

# **Введение**

Сортировка данных занимает важное место в современных компьютерных технологиях и является одной из наиболее востребованных процедур при обработке информации. Задачи сортировки встречаются практически во всех профессиональных областях, где требуется систематизация и быстрый поиск данных.

Алгоритмы сортировки составляют отдельный класс методов обработки информации. Они применяются для упорядочивания данных, что позволяет значительно ускорить последующий поиск и обработку. Сортировка служит отличным примером для изучения различных фундаментальных принципов построения алгоритмов, поскольку существует множество способов решения одной и той же задачи, каждый из которых обладает своими особенностями и преимуществами.

Пузырьковая сортировка - один из самых простых и наглядных алгоритмов сортировки. Её суть заключается в последовательном сравнении соседних элементов массива и обмене их местами, если они расположены в неправильном порядке. Этот процесс повторяется до тех пор, пока весь массив не будет отсортирован. Несмотря на свою простоту, пузырьковая сортировка демонстрирует основные принципы работы алгоритмов сортировки и часто используется в учебных целях для знакомства с базовыми алгоритмическими приёмами.[[4](#заклад4)]

Следует отметить, что пузырьковая сортировка уступает по эффективности более сложным алгоритмам, таким как быстрая сортировка или сортировка вставками. Однако благодаря своей простоте и наглядности, она остаётся популярной для обучения и решения задач с небольшими объёмами данных. Термин "сортировка" обычно означает процесс перестановки элементов множества в определённом порядке, и пузырьковая сортировка является классическим примером реализации этого процесса.

# **Постановка задачи**

Необходимо заполнить массив из n элементов случайными числами и записать исходные данные в отдельный файл. После этого требуется выполнить сортировку пузырьком над элементами массива, сохранить отсортированные данные во второй файл, а также определить время выполнения сортировки и количество перестановок элементов в процессе сортировки.

Для организации совместной работы над проектом используется сервис GitHub. Каждый участник выкладывает коммиты, отражающие его вклад и этапы выполнения задания. По итогам работы оформляется подробный отчет о проделанной практике.

## **Достоинства алгоритма сортировки пузырьком**

* алгоритм очень легко реализовать на любом языке программирования, а его принцип работы интуитивно понятен даже начинающим программистам;
* пузырьковая сортировка сохраняет относительный порядок элементов с одинаковыми значениями, что важно для некоторых задач обработки данных;
* для сортировки не требуется выделять дополнительную память, так как все операции происходят непосредственно внутри исходного массива.

## **Недостатки алгоритма сортировки пузырьком**

* алгоритм имеет квадратичную временную сложность O(n2), из-за чего работает очень медленно при увеличении объёма данных;
* даже если массив почти отсортирован, пузырьковая сортировка всё равно выполняет множество сравнений и обменов, что приводит к избыточным операциям и потере времени;
* из-за своей неэффективности пузырьковая сортировка практически не используется в реальных задачах, уступая место более быстрым алгоритмам, таким как быстрая сортировка или сортировка слиянием.

## **Типичные сценария применения данного алгоритма**

* формирование очереди на приём или обслуживание (в небольших компаниях или медицинских учреждениях администратор может вручную или с помощью простой программы сортировать список клиентов по времени записи, чтобы быстро определить порядок обслуживания);
* приоритизация задач в проектной или продуктовой команде (для небольших списков задач можно использовать пузырьковую сортировку, чтобы самые приоритетные задачи "всплывали" вверх списка и были видны в первую очередь, обеспечивая эффективное распределение ресурсов и времени);
* сортировка сотрудников по критериям для внутреннего анализа (упорядочить сотрудников по стажу работы, количеству выполненных проектов или другим показателям).

# **Выбор решения**

Для реализации данной программы также был выбран язык программирования C. Этот язык широко используется благодаря своей универсальности, эффективности и возможности работы как с низкоуровневыми, так и с высокоуровневыми задачами. C отличается высокой производительностью, экономным использованием ресурсов и хорошей переносимостью между различными платформами, что делает его оптимальным выбором для создания самых разных программных продуктов. [[2](#заклад2)]

В качестве среды разработки была использована Microsoft Visual Studio. Эта интегрированная среда предназначена для создания приложений под Windows и поддерживает как консольные, так и графические проекты, обеспечивая удобные инструменты для написания, отладки и тестирования программ.

Для удобства совместной работы над проектом мы использовали сразу несколько инструментов, каждый из которых отвечал за определённую часть командного взаимодействия.

Для управления версиями в проекте применялся **GitHub**, представляющий собой платформу, функционирующую на основе распределённой системы контроля версий **Git**. Он выступал в роли ядра для отслеживания изменений и ведения истории модификаций исходного кода.

Для оперативного обмена сообщениями и обсуждения текущих задач мы создали общий чат в **Telegram**. Это позволило быстро решать рабочие вопросы, делиться файлами и поддерживать постоянную связь между всеми участниками.

Для проведения голосовых обсуждений и созвонов использовался **Discord**. Эта платформа обеспечила удобную организацию онлайн-встреч, где можно было обсудить сложные моменты реализации, распределить задачи и принять коллективные решения.

# **Описание программы**

При запуске программы выводится меню из пяти пунктов:

1. создание массива с клавиатуры;
2. генерация случайного массива;
3. загрузка массива из файла;
4. сортировка текущего массива;
5. выход.

printf("\nМеню:\n");

printf("1. Ввести массив вручную\n");

printf("2. Сгенерировать случайный массив\n");

printf("3. Загрузить массив из файла\n");

printf("4. Отсортировать текущий массив\n");

printf("5. Выйти\n");

printf("Выберите действие: ");

Пользователю требуется выбрать нужный пункт меню.

При выборе пунктов 1–3 выводится сообщение, в котором пользователю необходимо ввести параметры для создания массива:

* при выборе пункта **1** — ввести целые числа через пробел или с новой строки, завершив ввод символом «!»; введённые значения сохраняются в файл (по умолчанию Array.txt);

while (token != NULL) {

if (strcmp(token, "!") == 0) {

fclose(file);

printf("Файл %s успешно создан.\n", filename);

getchar();

system("cls");

return 1;

}

int value;

if (sscanf\_s(token, "%d", &value) == 1) {

fprintf(file, "%d ", value);

}

token = strtok\_s(NULL, " \n", &context);

* при выборе пункта **2** — ввести размер массива и диапазон случайных чисел и передается в функцию, которая генерирует массив, записывает его в файл и выводит на экран;

int MakeArrayFile(int min, int max, int size, const char\* filename) {

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

srand(time(NULL));

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

int num = rand() % (max - min + 1) + min;

printf("%d ", num);

fprintf(file, "%d ", num);

}

fclose(file);

printf("\n");

return 1;

}

* при выборе пункта **3** — ввести имя файла, из которого будет загружен массив.

printf("Введите имя файла: ");

char filename\_buffer[100];

fgets(filename\_buffer, sizeof(filename\_buffer), stdin);

filename\_buffer[strcspn(filename\_buffer, "\n")] = '\0';

if (current\_array) free(current\_array);

if (readArrayFromFile(filename\_buffer, &current\_array, &size)) {

strcpy\_s(current\_filename, sizeof(current\_filename), filename\_buffer);

printf("Массив успешно загружен. Размер: %d\n", size);

}

После создания или загрузки массива пользователь может выбрать пункт **4** - сортировка текущего массива.

printf("Тип сортировки:\n1. По убыванию\n2. По возрастанию\nВыберите: ");

…

bubbleSort(current\_array, size, method);

Программа предлагает выбрать тип сортировки:

* по возрастанию;
* по убыванию.

Для сортировки используется метод пузырька: массив перебирается слева направо, сравниваются соседние элементы и меняются местами, если они расположены не в нужном порядке. Этот процесс повторяется, пока массив не будет полностью отсортирован.

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (method == 2) {

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

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

else if (method == 1) {

if (arr[j] < arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

После сортировки отсортированный массив выводится на экран и записывается в файл SortedArray.txt.

void RecordSortedArray(const char\* filename, int\* array, int size) {

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

if (file == NULL) {

printf("Ошибка: не удалось создать файл %s\n", filename);

return;

}

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

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

}

fclose(file);

printf("Массив успешно записан в файл %s\n", sorted\_filename);

}

Также отображается время выполнения сортировки и количество перестановок.

При выборе пункта **5** программа завершает выполнение, предварительно освобождая выделенную память.

Подробный алгоритм работы программы ([рис. 1](#рис1)) и функции сортировки ([рис. 2](#рис2)) представлен в подразделе 4.1, 4.2.

Листинг программы приведен в приложении A.

# **Схемы программы**

# **Блок-схема программы**

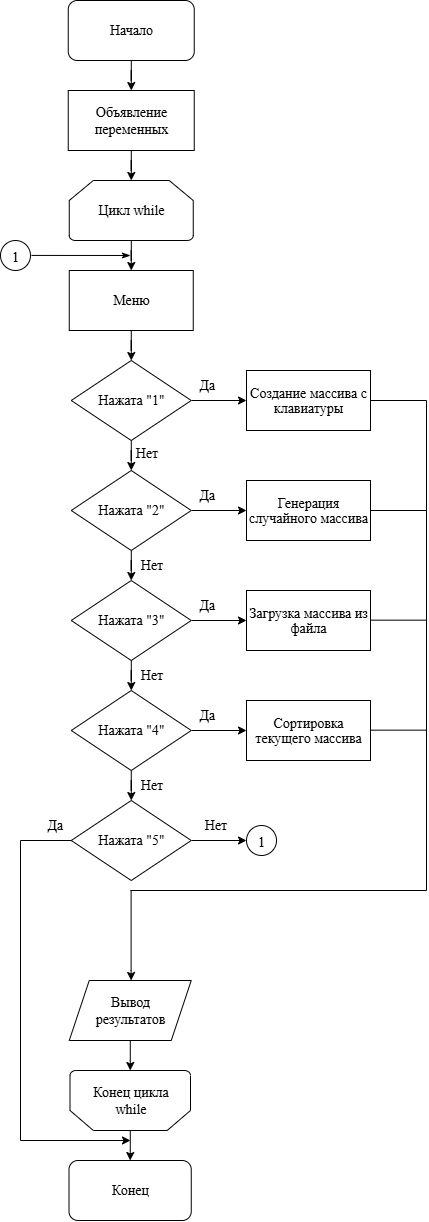


Рисунок 1 -Блок-схема программы

# **Блок-схема алгоритма**

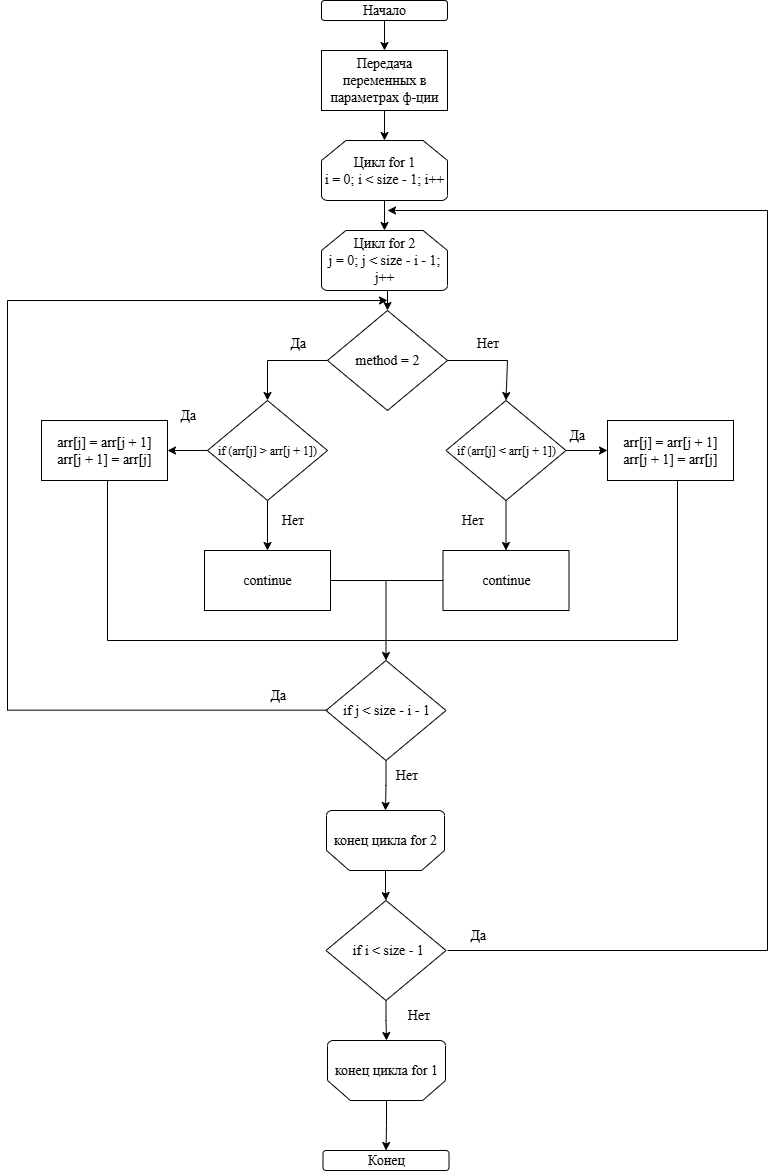


Рисунок 2 -Блок-схема алгоритма

# **Тестирование программы**

Тестирование ([рис. 3](#рис3)) показало, что с увеличением количества элементов пропорционально увеличивается время работы программы.

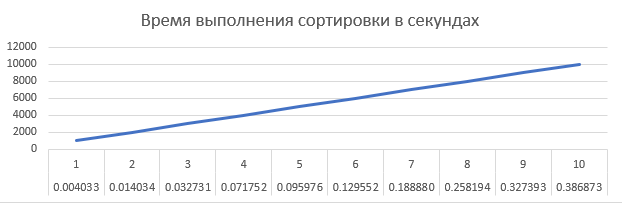


Рисунок 3 -Результаты тестирования

# **Отладка**

В качестве среды разработки для реализации и отладки программы была выбрана Microsoft Visual Studio, так как она предоставляет все необходимые инструменты для написания, компиляции и тестирования программ на языке C.

В процессе отладки программы активно использовались точки останова (breakpoints) и пошаговое выполнение кода. Точки останова позволяют приостановить выполнение программы в нужном месте и перейти к просмотру состояния переменных и стека вызовов. [[3](#заклад3)] Это дает возможность детально проанализировать выполнение программы на каждом этапе и выявить возможные ошибки.

Для поиска и устранения ошибок в программе применялся встроенный отладчик Visual Studio. С его помощью можно исследовать значения локальных и глобальных переменных, отслеживать изменения данных в массивах, а также контролировать ход выполнения алгоритмов, таких как сортировка пузырьком.

Для ускорения поиска ошибок использовался метод локализации проблемы путем поэтапной проверки отдельных участков кода. Такой подход позволяет быстрее определить, на каком этапе возникает некорректное поведение программы.

В ходе отладки также применялся просмотр содержимого переменных и массивов непосредственно в окне отладчика. Это позволяло в реальном времени отслеживать изменения данных, выявлять некорректные значения и анализировать логику работы программы без необходимости добавления дополнительных операторов вывода.

# **Совместная разработка**

Для удобства совместной разработки был использован сервис GitHub (Projects) ([рис. 4](#рис4)). Определили задачи проекта, назначили приоритет задачам, разделили роли.

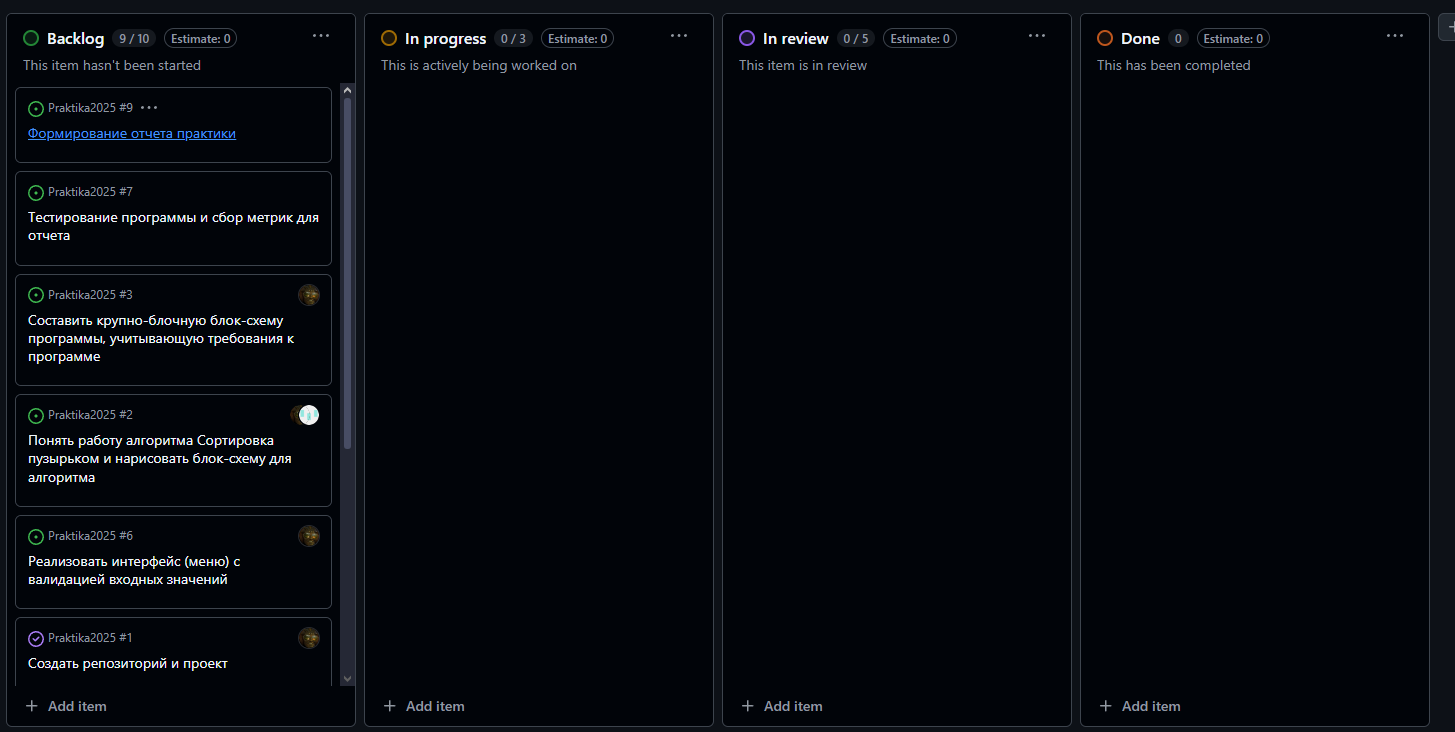


Рисунок 4 -Определение задач проекта

Обсуждали и советовались ([рис. 5](#рис5)) в общей беседе.



Рисунок 5 -Полезная литература

Корректировали статус задач по мере выполнения ([рис. 6](#рис6)).

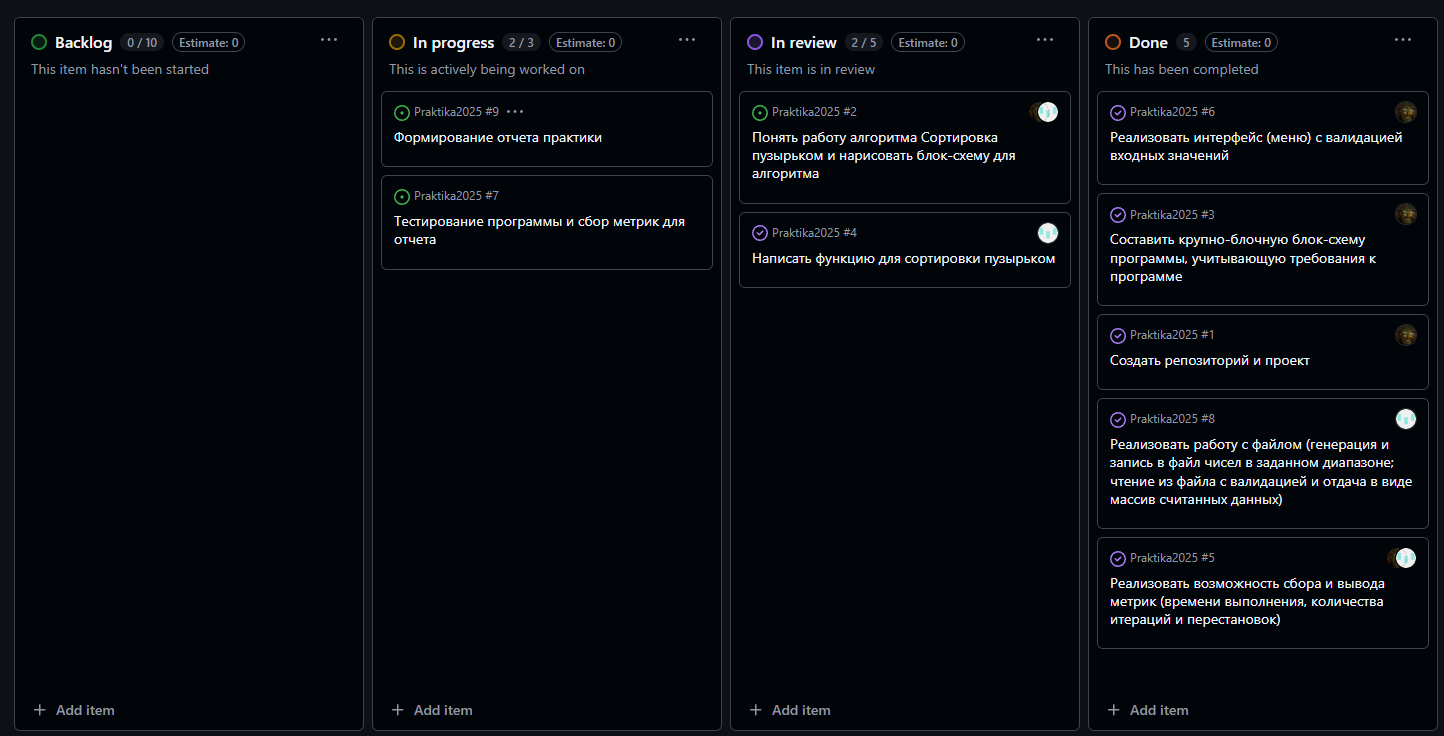


Рисунок 6 -Канбан доска

В рамках работы над проектом я инициализировал ([рис. 7](#рис7)) репозиторий, а затем склонировал ([рис. 8](#рис8)) его копию с удалённого сервера для локальной работы. В процессе выполнения данной практики наша бригада осуществляла совместную работу в GitHub, что обеспечивало эффективную координацию действий и интеграцию изменений каждого участника в общий проект.

Мной было реализован интерфейс программы, организовал валидацию вводимых пользователем символов, сохранение массива в файл и чтение массива из файла. Позже код был загружен на ветку «interface» ([рис. 9](#рис9)), с последующим слиянием с веткой «main».



Рисунок 7 -Инициализация репозитория

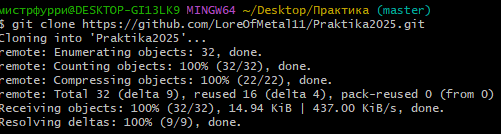


Рисунок 8 – Загрузка папки с удаленного репозитория

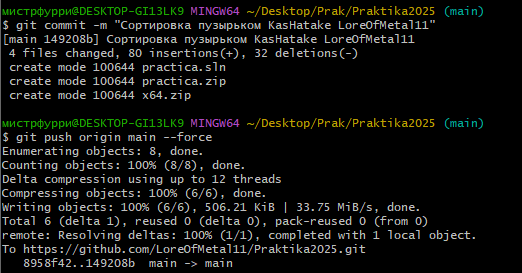


Рисунок 9 – Загрузка программы на удаленный репозиторий

Ссылка на удаленный репозиторий:

<https://github.com/LoreOfMetal11/Praktika2025>

# **Заключение**

В ходе выполнения данной работы были приобретены практические навыки разработки программ на языке C с использованием среды Microsoft Visual Studio. Был реализован консольный интерфейс для работы с одномерными массивами, обеспечена валидация пользовательского ввода, а также реализованы функции создания, генерации, загрузки, сортировки и сохранения массивов.  
Особое внимание уделялось корректной работе с файлами и поддержке кириллицы в консольном приложении.

В процессе работы был подробно изучен и реализован алгоритм сортировки пузырьком с возможностью выбора порядка сортировки. Программа была протестирована, а исходный код загружен в удалённый репозиторий на GitHub, что позволило освоить основы работы с системами контроля версий.

Выполнение данной работы способствовало развитию навыков структурирования кода, отладки и тестирования программ, а также работе с динамической памятью и пользовательским вводом.

В дальнейшем программу можно усовершенствовать за счёт добавления других алгоритмов сортировки, расширения функционала пользовательского интерфейса и интеграции с графическими библиотеками для повышения удобства работы.

# **Список используемой литературы**

1. ГОСТ 19.701 - 90 Схемы алгоритмов, программ, данных и систем;
2. Гриффитс Д., Гриффитс Д. Изучаем программирование на С. – СПб.: Питер, 2014;
3. Программирование на языке C [Электронный ресурс]. – URL: [https://metanit.com/cpp/c/](https://metanit.com/cpp/c/%20) (дата обращения: 30.06.2025 г.);
4. Как работает пузырьковая сортировка [Электронный ресурс]. – URL: [https://thecode.media/bubble-sort (дата обращения 29.06.2025).](https://thecode.media/bubble-sort/)

# **Приложение А. Листинг программы**

#include <stdio.h>

#include <time.h>

#include <string.h>

#include <stdlib.h>

#include <windows.h>

#include <ctype.h>

const char\* default\_filename = "Array.txt";

const char\* sorted\_filename = "SortedArray.txt";

typedef struct {

double time\_spent; // время прогона, до миллионых у нас

int swapCount; // количество перестановок, добавили

} SortResult;

int readArrayFromFile(const char\* filename, int\*\* array, int\* size) {

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

if (file == NULL) {

printf("Ошибка: не удалось открыть файл %s\n", filename);

return 0;

}

int count = 0;

int temp;

while (fscanf\_s(file, "%d", &temp) == 1) count++;

if (count == 0) {

printf("Файл пуст или содержит нечисловые данные\n");

fclose(file);

return 0;

}

\*array = (int\*)malloc(count \* sizeof(int));

if (\*array == NULL) {

printf("Ошибка выделения памяти\n");

fclose(file);

return 0;

}

rewind(file);

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

if (fscanf\_s(file, "%d", &(\*array)[i]) != 1) {

printf("Ошибка чтения числа из файла\n");

free(\*array);

fclose(file);

return 0;

}

}

fclose(file);

\*size = count;

return 1;

}

void RecordSortedArray(const char\* filename, int\* array, int size) {

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

if (file == NULL) {

printf("Ошибка: не удалось создать файл %s\n", filename);

return;

}

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

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

}

fclose(file);

printf("Массив успешно записан в файл %s\n", filename);

}

SortResult bubbleSort(int\* arr, int size, int method) {

LARGE\_INTEGER frequency, start, end;

QueryPerformanceFrequency(&frequency);

QueryPerformanceCounter(&start);

SortResult result = { 0 }; // илья, тут структуры, посмотри пожалуйста. мы сделали

result.swapCount = 0;

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (method == 2) {

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

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

result.swapCount++;

}

}

else if (method == 1) {

if (arr[j] < arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

result.swapCount++;

}

}

}

}

QueryPerformanceCounter(&end);

result.time\_spent = (double)(end.QuadPart - start.QuadPart) / frequency.QuadPart;

return result;

}

int MakeArrayFile(int min, int max, int size, const char\* filename) {

if (min >= max) {

printf("Ошибка: нижний предел должен быть меньше верхнего.\n");

return 0;

}

if (size <= 0) {

printf("Ошибка: размер массива должен быть положительным.\n");

return 0;

}

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

if (file == NULL) {

printf("Ошибка: не удалось создать или открыть файл %s.\n", filename);

return 0;

}

srand(time(NULL));

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

int num = rand() % (max - min + 1) + min;

printf("%d ", num);

fprintf(file, "%d ", num);

}

fclose(file);

printf("\n");

printf("Файл %s успешно создан.\n", filename);

return 1;

}

int OwnArray(const char\* filename) {

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

if (file == NULL) {

printf("Ошибка открытия файла!\n");

return 0;

}

printf("Введите целые числа через пробел или каждое с новой строки.\n");

printf("Для завершения ввода введите '!' на новой строке:\n");

char input[256];

char\* context = NULL;

while (1) {

fgets(input, sizeof(input), stdin);

if (strchr(input, '!') != NULL) break;

char\* token = strtok\_s(input, " \n", &context);

while (token != NULL) {

if (strcmp(token, "!") == 0) {

fclose(file);

printf("Файл %s успешно создан.\n", filename);

getchar();

system("cls");

return 1;

}

int value;

if (sscanf\_s(token, "%d", &value) == 1) {

fprintf(file, "%d ", value);

}

token = strtok\_s(NULL, " \n", &context);

}

}

fclose(file);

printf("Файл %s успешно создан.\n", filename);

return 1;

}

int checkForExtraChars() {

int c = getchar();

return (c != '\n' && c != EOF);

}

int main() {

SetConsoleCP(1251); // UTF-8 а то у денисыча не отображаются символы кириллицы

SetConsoleOutputCP(1251);

system(cls);

srand(time(NULL));

char current\_filename[100] = "Array.txt";

int size = 0;

int\* current\_array = NULL;

while (1) {

printf("\nМеню:\n");

printf("1. Создать массив с клавиатуры\n");

printf("2. Сгенерировать случайный массив\n");

printf("3. Загрузить массив из файла\n");

printf("4. Отсортировать текущий массив\n");

printf("5. Выйти\n");

printf("Выберите действие: ");

int choice;

if (scanf\_s("%d", &choice) != 1) {

printf("Ошибка ввода. Пожалуйста, введите число.\n");

while (getchar() != '\n');

continue;

}

switch (choice) {

case 1: {

while (getchar() != '\n');

printf("Введите имя файла (по умолчанию %s): ", current\_filename);

char filename\_buffer[100];

fgets(filename\_buffer, sizeof(filename\_buffer), stdin);

filename\_buffer[strcspn(filename\_buffer, "\n")] = '\0';

if (strlen(filename\_buffer) > 0) {

strcpy\_s(current\_filename, sizeof(current\_filename), filename\_buffer);

}

if (OwnArray(current\_filename)) {

if (current\_array) free(current\_array);

if (readArrayFromFile(current\_filename, &current\_array, &size)) {

printf("Массив успешно загружен. Размер: %d\n", size);

}

}

printf("Нажмите Enter чтобы продолжить.\n");

getchar();

system("cls");

break;

}

case 2: {

printf("Введите размер массива: ");

if (scanf\_s("%d", &size) != 1 || size <= 0 || checkForExtraChars()) {

printf("Неверный размер массива.\n");

while (getchar() != '\n');

continue;

}

int min, max;

printf("Введите диапазон (min max): ");

if (scanf\_s("%d %d", &min, &max) != 2 || min >= max || checkForExtraChars()) {

printf("Неверный диапазон.\n");

while (getchar() != '\n');

continue;

}

if (MakeArrayFile(min, max, size, current\_filename)) {

if (current\_array) free(current\_array);

if (readArrayFromFile(current\_filename, &current\_array, &size)) {

printf("Массив успешно создан и загружен. Размер: %d\n", size);

}

}

printf("Нажмите Enter чтобы продолжить.\n");

getchar();

system("cls");

break;

}

case 3: {

while (getchar() != '\n');

printf("Введите имя файла: ");

char filename\_buffer[100];

fgets(filename\_buffer, sizeof(filename\_buffer), stdin);

filename\_buffer[strcspn(filename\_buffer, "\n")] = '\0';

if (current\_array) free(current\_array);

if (readArrayFromFile(filename\_buffer, &current\_array, &size)) {

strcpy\_s(current\_filename, sizeof(current\_filename), filename\_buffer);

printf("Массив успешно загружен. Размер: %d\n", size);

}

printf("Нажмите Enter чтобы продолжить.\n");

getchar();

system("cls");

break;

}

case 4: {

if (!current\_array || size == 0) {

printf("Нет загруженного массива для сортировки.\n");

break;

}

printf("Тип сортировки:\n1. По убыванию\n2. По возрастанию\nВыберите: ");

int method;

if (scanf\_s("%d", &method) != 1 || (method != 1 && method != 2) || checkForExtraChars()) {

printf("Неверный выбор.\n");

while (getchar() != '\n');

continue;

}

SortResult sortResult = bubbleSort(current\_array, size, method);

printf("Отсортированный массив:\n");

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

printf("%d ", current\_array[i]);

}

printf("\nВремя выполнения: %.6f секунд\n", sortResult.time\_spent);

printf("Количество перестановок: %d\n", sortResult.swapCount);

RecordSortedArray(sorted\_filename, current\_array, size);

printf("Нажмите Enter чтобы продолжить.\n");

getchar();

system("cls");

break;

}

case 5:

system("cls");

printf("\n\n\n\n==========================\nРабота программы завершена.\nЖелаем хорошего дня.\n==========================\n\n\n\n"); // желаем хорошего дня, чтобы нам поставили топовую оценочку -денис, ярик

if (current\_array) free(current\_array);

return 0;

default:

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

}

}

}