

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 ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Арцов Артём Алексеевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация
автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных
машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., 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

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Арцов Артём Алексеевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., 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	26	20.06.25 – 24.06.25	
2	Подбор и изучение материала по теме работы	26	24.06.25 – 26.06.25	
3	Разработка алгоритма	26	26.06.25 – 28.06.25	
4	Описание алгоритма и программы	30	04.07.25 – 09.07.25	
5	Тестирование	30	09.07.25 – 12.07.25	
6	Получение и анализ результатов	38	12.07.25 – 14.07.25	
7	Оформление отчёта	40	14.07.25 – 17.07.25	
	Общий объём часов	216		

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)

Арцов Артём Алексеевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 года

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Арцов А.А. выполнял практическое задание «Сортировка пузырьком». На первоначальном этапе были изучен и проанализирован алгоритм пузырьковой сортировки, был выбран метод решения и язык программирования C, на котором была написана программа по реализации сортировки массива методом пузырька, также, осуществлена реализация работы с файлами.

Специалист Арцов А.А. "___" _____ 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

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Арцов Артём Алексеевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

В процессе выполнения практики Арцов А.А. решал следующие задачи: разработка алгоритма быстрой сортировки, анализ результатов сортировки, сравнение существующих методов с разработанным алгоритмом.

За период прохождения практики были освоены основные понятия и технологии программирования, изучены основные инструменты языков C/C++. И получены следующие результаты: разработан алгоритм быстрой сортировки, получены результаты работы алгоритма, сделаны выводы эффективности работы алгоритма быстрой сортировки. Во время выполнения работы Арцов А.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.

За выполнение работы Арцов А.А. заслуживает оценки « ».

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

Содержание

Введение	2
1 Постановка задачи	3
1.1 Достоинства алгоритма сортировки пузырьком	3
1.2 Недостатки алгоритма сортировки пузырьком	3
1.3 Типичные сценарии применения данного алгоритма	4
2 Выбор решения	5
3 Описание программы	6
3.1 Описание личного вклада в проект	7
4 Схемы программы	9
4.1 Блок-схема алгоритма	9
4.2 Блок-схемы программы	10
5 Тестирование программы	16
6 Отладка	17
7 Совместная работа	18
Заключение	23
Список используемой литературы	24
Приложение А. Листинг программы	25

Введение

В наше время сортировка данных играет очень важную роль и является одним из главных процессов обработки данных на текущий момент. В контексте профессиональной деятельности очень часто можно встретить задачу, которая подразумевает сортировку данных различными способами и методами.

Самых методов и алгоритмов сортировки достаточно много. Применяются они с целью осуществления ускорения и упрощения поиска данных.

Сама важность и необходимость сортировки заключается в том, что на её примере можно достаточно наглядно показать принцип работы определённых алгоритмов. При всём этом, алгоритмов сортировки очень много и каждый имеет свои преимущества и свои недостатки относительно друг друга. Так, в определённых ситуациях можно применить простой алгоритм и этого будет более чем достаточно, а иногда нужно применить достаточно сложный алгоритм в достижении нужного результата. Сама цель сортировки заключается в перестановке некоторого количества объектов в определённом порядке по определённым правилам.

Сортировка пузырьком — это один из квадратичных алгоритмов сортировки. Основное преимущество этого метода заключается в простоте его реализации и минимальных требованиях к памяти. Но данный метод имеет недостаток, который виден при работе с большими массивами и объёмами данных, при этом будет наблюдаться низкая скорость работы алгоритма. Такая сортировка очень хороший пример для обучения сортировке, но не является оптимальным методом в реальных приложениях с большим объёмом данных.

1 Постановка задачи

Поставленная задача: необходимо заполнить массив из n -ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить сортировку методом пузырька над этими данными, которые находятся в массиве, записать отсортированные данные в другой файл, затем посчитать время выполнения и количество перестановок значений массива при сортировке.

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

Оформить отчет по проведенной практике.

1.1 Достоинства алгоритма сортировки пузырьком

- не используется дополнительная память;
- для маленьких массивов предпочтительнее, чем другие алгоритмы;
- легкость реализации. Алгоритм очень прост для понимания и реализации на любом языке программирования.

1.2 Недостатки алгоритма сортировки пузырьком

- низкая эффективность. Сложность алгоритма составляет $O(n^2)$, что делает его крайне медленным для работы с большими массивами данных;
- ресурсоёмкий по числу операций;
- для заранее отсортированных данных все равно выполняется несколько лишних проходов, что делает алгоритм не эффективным в реальных условиях.

1.3 Типичные сценарии применения данного алгоритма

- обучение основам алгоритмизации и программирования в образовательных целях;
- стандартная ситуация в быте (раскладка книг, документов или файлов по алфавиту или дате);
- создание наглядных анимаций, которые демонстрируют принципы работы сортировки (например, в визуализаторах алгоритмов или обучающих роликах).

2 Выбор решения

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. Си – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью и переносимостью. Указанные преимущества Си обеспечивают хорошее качество разработки почти любого вида программного продукта.

В качестве среды программирования была выбрана программа *Microsoft Visual Studio Code*. *Microsoft Visual Studio Code* — это редактор кода для разных языков программирования.

Для удобства совместной разработки был использован сервис *WEEEEK*. *WEEEEK* — сервис для управления личными и командными проектами. В основе *WEEEEK* лежит недельный планер и канбан-методология: доски, колонки и т. д. Проект динамично разрабатывается, регулярно расширяя функционал и возможности. Ведется активная работа с пожеланиями пользователей в еженедельном патчноте *WEEEEK*.

3 Описание программы

При запуске программы пользователю выводится меню с действиями:

- a) 1. Сгенерировать числа и записать в файл;
- b) 2. Отсортировать числа из файла;
- c) 3. Проверить сортировку чисел;
- d) 4. Сохранить отсортированные числа;
- e) 0. Выход.

Пользователю требуется выбрать тот пункт, который ему требуется. При выборе пунктов числа 1-4 и 0 являются ключами к действиям, т.е. нажимая одну из цифр на клавиатуре пользователь выберет нужное ему действие и после выбора определённого пункта, выводится сообщение, в котором пользователю необходимо ввести определенные данные.

После того как пользователь ввёл название файла, количество чисел и диапазон значений, массив с числами генерируется в файл (в данном проекте файл может быть *txt* или *csv*). Название файла и расширение пользователь задаёт сам. После эти данные читаются из файла.

Затем этим данные сортируются методом пузырька по возрастанию или по убыванию. Происходит это путём сравнения соседних элементов. Если текущий элемент больше чем следующий стоящий, то происходит перестановка, т.е. текущий встаёт на место следующего, а тот что был впереди становится сзади, т.о. самые большие элементы «всплывают» в конец массива. Действия повторяются пока массив не будет отсортирован.

Далее пользователь может проверить, а отсортирован ли массив, текстовые сообщения в консоли дадут ему понять, в каком состоянии данные. После этого можно сохранить отсортированный массив в файл, который пользователь также сам задаёт.

На цифру 0 можно завершить выполнение программы.

Подробный алгоритм будет приложен в подразделе 4.1 на рисунке 1.

3.1 Описание личного вклада в проект

Моя задача заключалась в проектировании и реализации алгоритма сортировки пузырьком (без использования стандартных/сторонних библиотек), а также реализовать обработку входных и выходных файлов.

Для записи и генерации чисел в файл была разработана функция *gen_file*, которая открывает файл для записи, записывает *count* (случайные числа) из диапазона от *min* до *max*.

```
for (int i = 0; i < count; i++) {  
    long big_rand = ((long)rand() << 15) | rand();  
    int nums = (int)(min + big_rand % (max - min + 1));  
    fprintf(file, "%d\n", nums);  
}
```

Это самый важный фрагмент кода, так как здесь происходит генерация числа в массив, а затем запись в файл. В цикле с установленным числом повторений происходит генерация и запись в файл. *Long big_rand* этой переменной присваивается достаточно важное действие. Проблема в том, что стандартная функция *rand()* возвращает число в диапазоне *0..RAND_MAX* (32767, а программа допускает 100 000 значений), что ограничивает «случайность» генерации, для этого было реализовано *rand() << 15*, что сдвигает биты первого *rand()* на 15 позиций влево (что приравнивается к умножению на 32768). *| rand()* — происходит объединение с новым *rand()*, заполняя младшие 15 бит. В результате получается число с типом *long*, которое имеет больший диапазон.

Для чтения из файла данных была разработана функция *read_file*, которая считывает введённые числа из файла в массив для дальнейшей работы с ними.

```

while (*count < MAX && fscanf(file, "%d",
&numbers[*count]) == 1) {
    (*count)++;
}

```

В данном цикле с предусловием, что пока число считанных чисел меньше максимально допустимого и пока чтение успешно, функция считывает числа из файла и складывает в массив *numbers*. Внутри цикла происходит увеличение счётчика с каждым удачным считыванием.

Для реализации самого алгоритма сортировки написана функция *bubble_sort*, которая проходит по всем числам сортируя массив из введенных чисел по возрастанию. Перестановка происходит с помощью временной переменной *tmp*, именно с помощью неё мы сможем переставить соседние элементы если они стоят не правильном порядке, именно здесь происходит сортировка методом пузырька. Затем происходит увеличение счетчика количества перестановок.

```

int tmp = array[j];
array[j] = array[j + 1];
array[j + 1] = tmp;
(*swap_count)++;

```

Также для случая, когда пользователь мог забыть нажать кнопку сортировки предусмотрена функция *check*, которая проверяет отсортирован ли массив. Она не несёт особой задачи в реализации алгоритма, но поможет пользователю, если он случайно допустил ошибку при выборе пунктов в меню.

Листинг программы приведён в приложении А.

4 Схемы программы

4.1 Блок-схема алгоритма

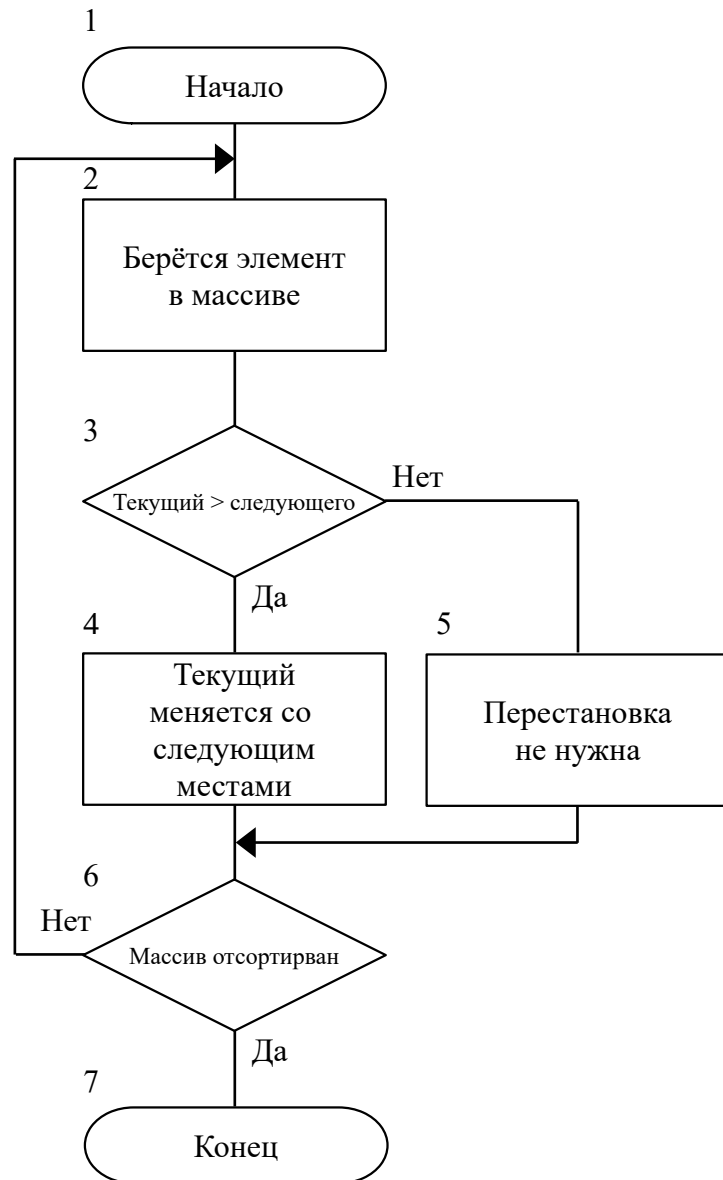
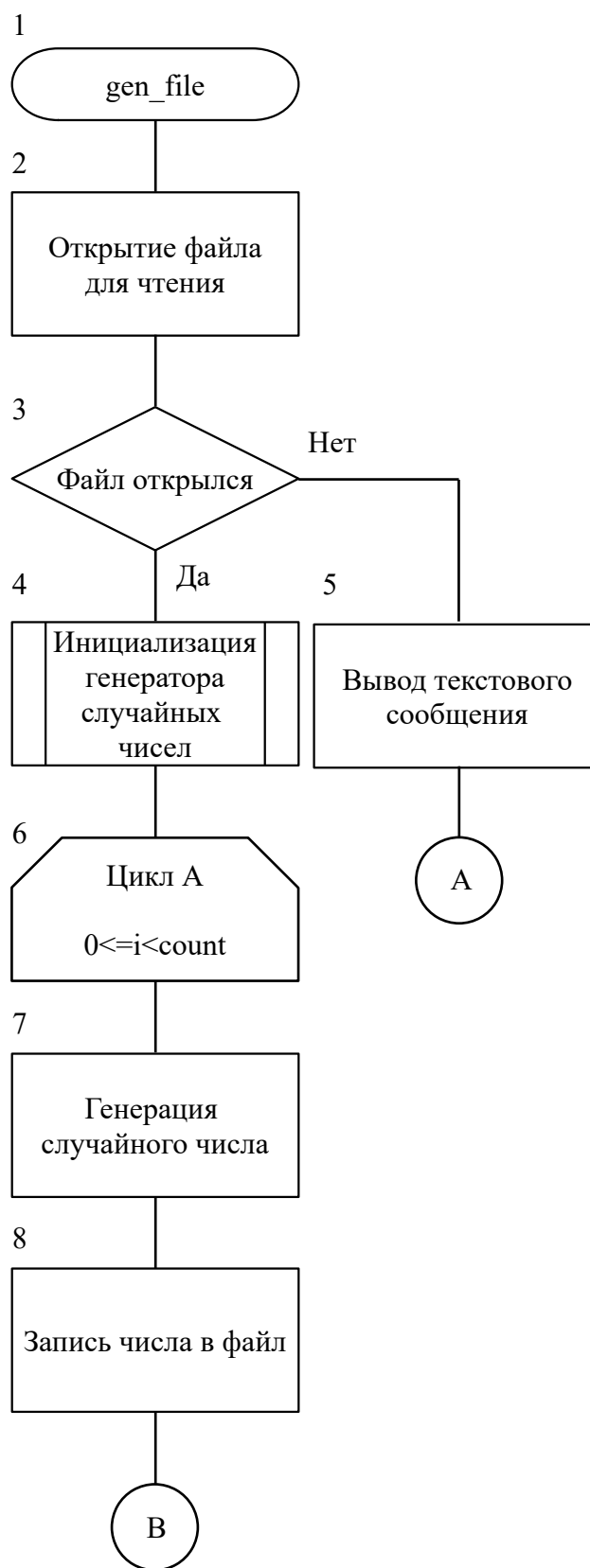


Рисунок 1 — Блок-схема алгоритма

4.2 Блок-схемы программы



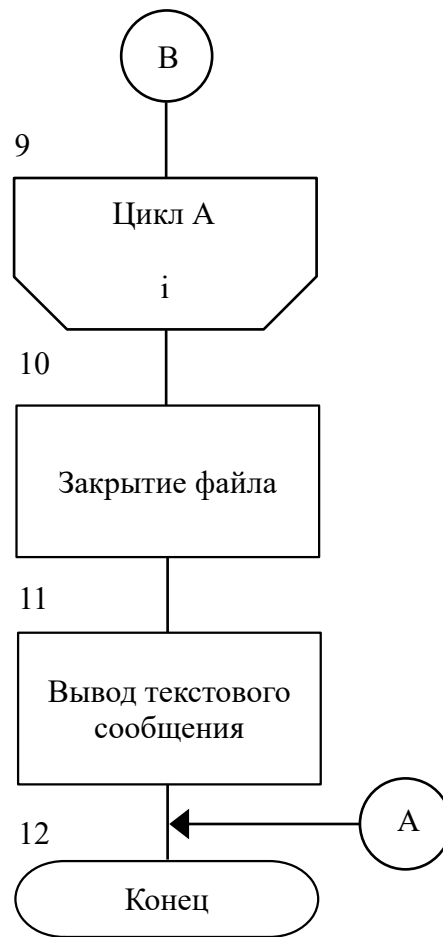
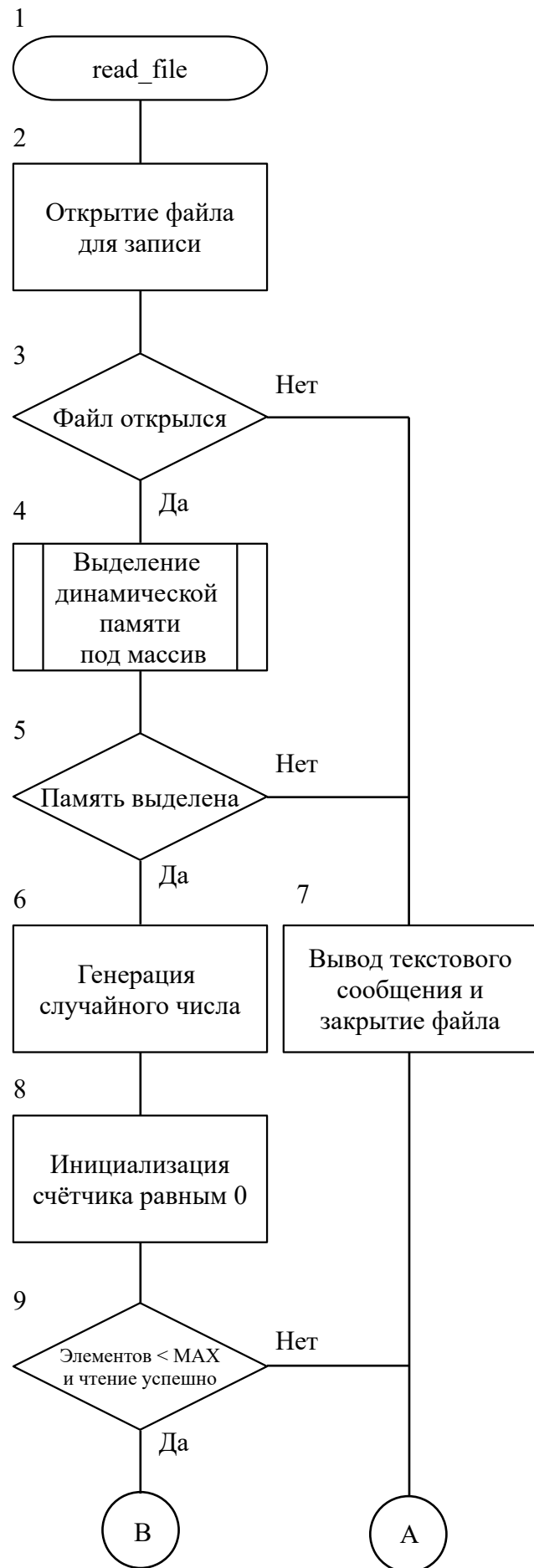


Рисунок 2 — Блок-схема функции *gen_file*



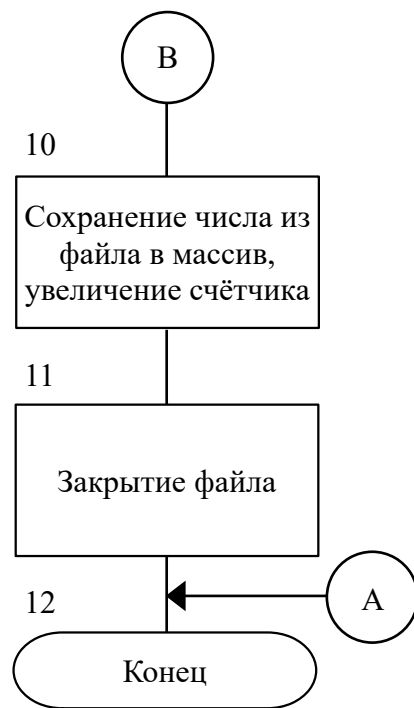


Рисунок 3 — Блок-схема функции *read_file*

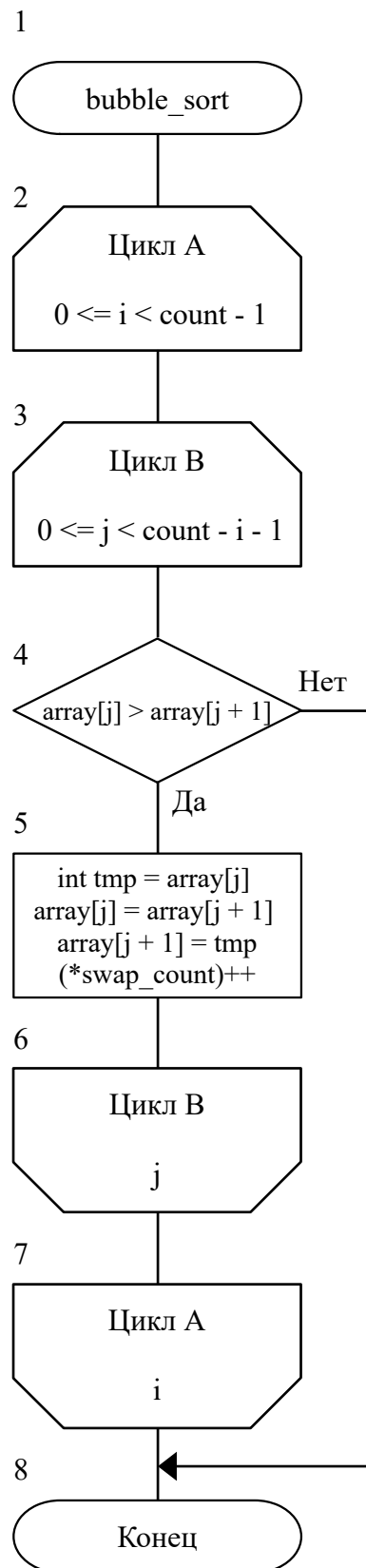


Рисунок 4 — Блок-схема функции *bubble_sort*

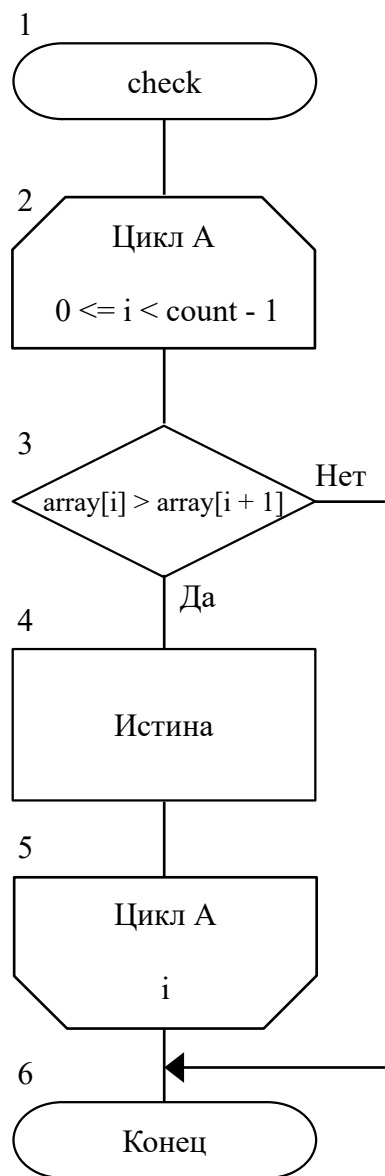


Рисунок 5 — Блок-схема функции *check*

5 Тестирование программы

Тестирование показало, что с увеличением количества элементов пропорционально увеличивается время выполнения программы. График зависимости времени выполнения сортировки от количества элементов в наборе приведен на рисунке 6.

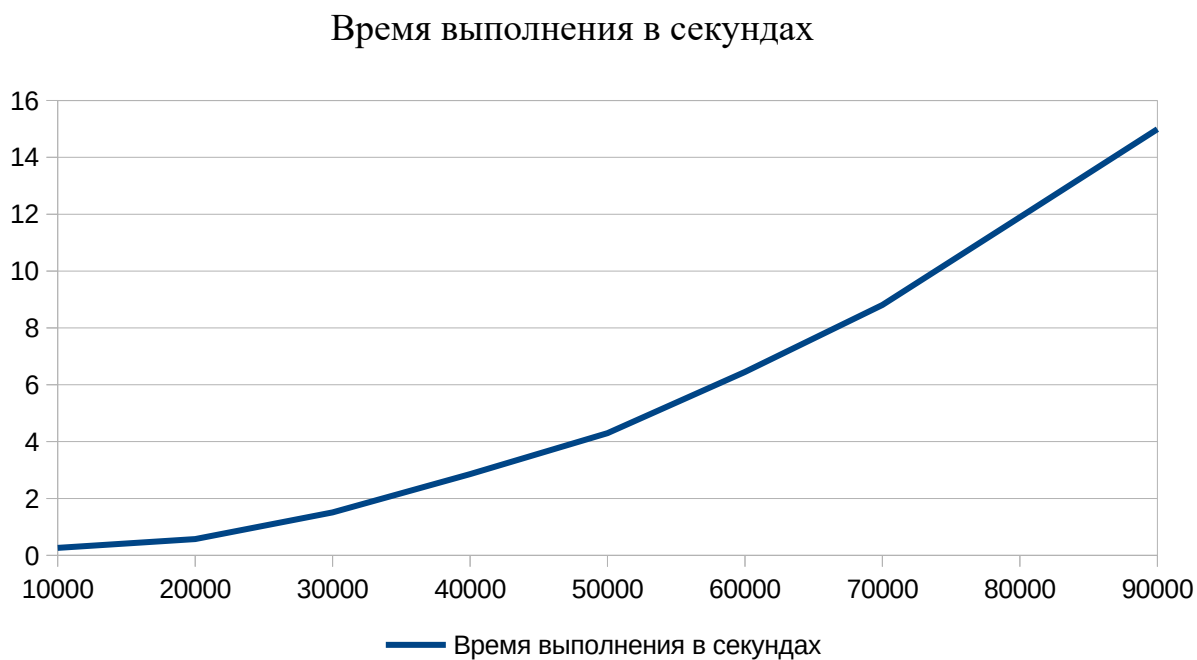


Рисунок 6 — Результаты тестирования

6 Отладка

В качестве среды разработки была выбрана программа *Microsoft Visual Studio Code*, которая содержит в себе необходимый функционал для разработки и отладки модулей и программы.

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

Точка останова (*breakpoint*) - это специальное место в коде программы, где отладчик временно приостанавливает выполнение, чтобы разработчик мог проанализировать состояние программы, проверить значения переменных, отследить ход выполнения этой самой программы.

Проверялись фрагменты кода, с помощью размещения точек останова: в начале *main()* для контроля входа в программу; перед вызовом *gen_file()* и *read_file()*; внутри циклов сортировки *bubble_sort* и *rev_bubble_sort*; также перед проверкой результатов в функциях *check* и *check1*.

При проведении отладки использовались команды такие как: (*Step Over*) - для обхода вызовов стандартных функций; (*Step Into*) - для входа в пользовательские функции; (*Step Out*) - для выхода из текущей функции.

7 Совместная разработка

Для удобства совместной разработки был использован сервис *WEEEK*.
Определили задачи проекта, назначили приоритет задачам.

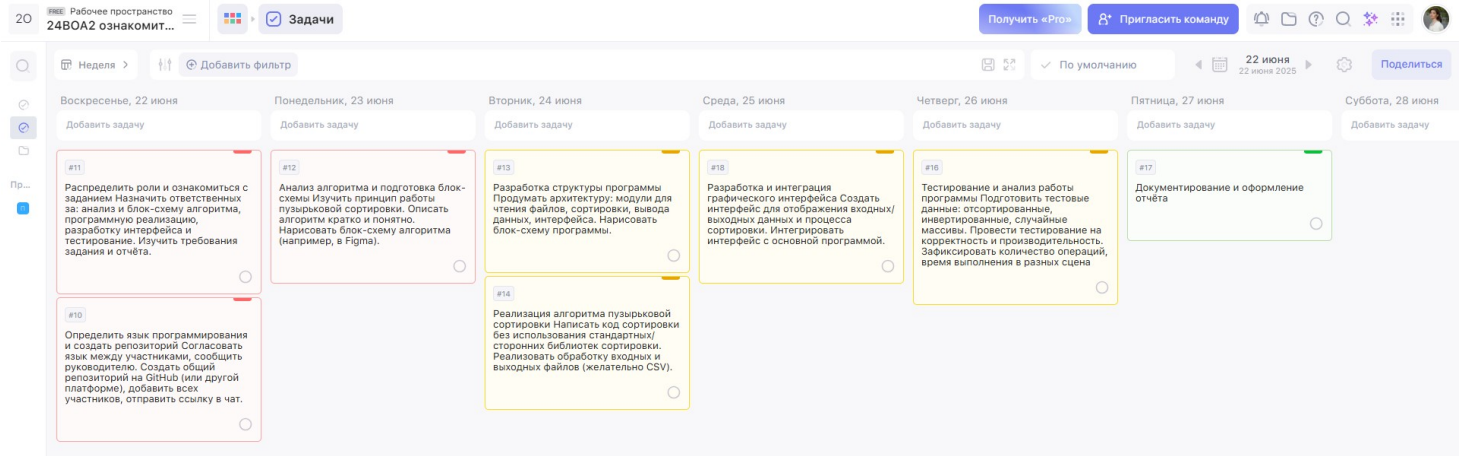


Рисунок 7 — Определение задач проекта

Затем разделили роли, назначили исполнителей задачам.

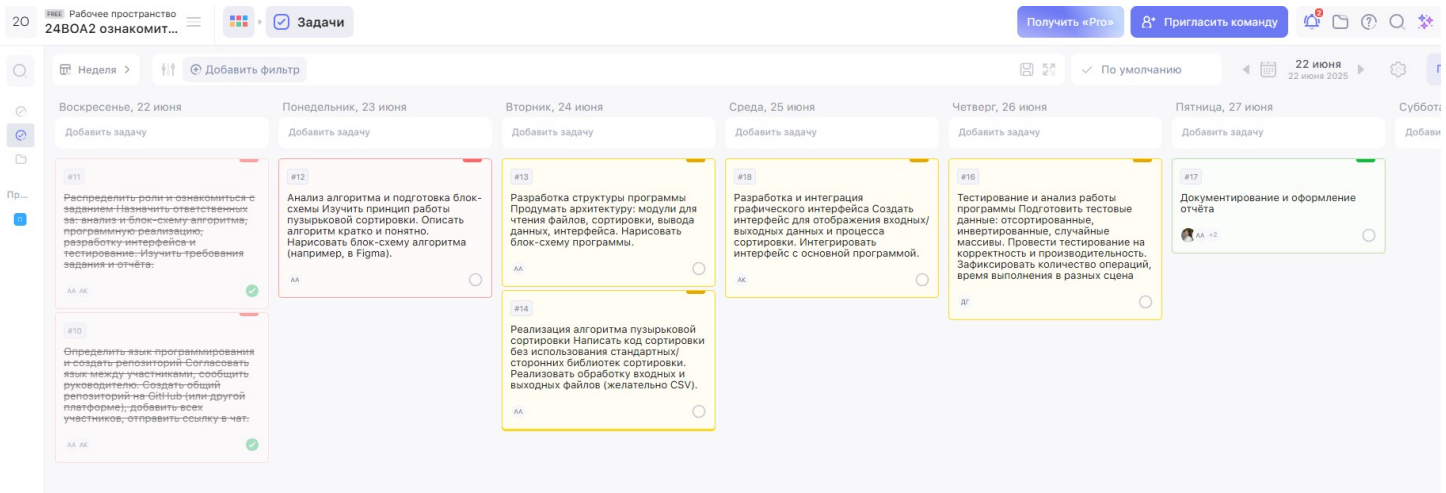


Рисунок 8 — Распределение задач проекта

20

РАБОЧЕЕ ПРОСТРАНСТВО
24BOA2 ознакомит...

Пользователи

Получить «Pro»

Пригласить команду

📧

🔍

⚙️

👤

ЛЮДИ

Команды

О сервисе

Мой профиль

Настройки профиля

📅

🔍

⚙️

Добавить фильтр

🔍 Поиск

✓ По умолчанию

Пригласить участников






Имя	Telegram	Почта аккаунта	Почта для связи	Должность	Роль	Доступы	Активность	Дата добавления	
 Эльмира Юмаева	-	elimirage@yandex.ru	-	-	Супер-админ		22.06.2025 09:31	22.06.2025	<div>Редактировать</div>
AA Артём Арцов	-	artemartsov@yandex.ru	-	программная реализация	Участник		22.06.2025 13:32	22.06.2025	<div>Редактировать</div> <div>Исключить</div>
AK Артём Курицын	-	codyz1@yandex.ru	-	графический интерфейс	Участник		22.06.2025 13:18	22.06.2025	<div>Редактировать</div> <div>Исключить</div>
ДГ Денис Гераськин	-	Klodinskiy@yandex.ru	-	тестирование программы	Участник		22.06.2025 13:27	22.06.2025	<div>Редактировать</div> <div>Исключить</div>

Рисунок 9 — Участники *WEEK*

Обсуждение, выполнение и корректировка работы проводилась на площадке *Discord*.

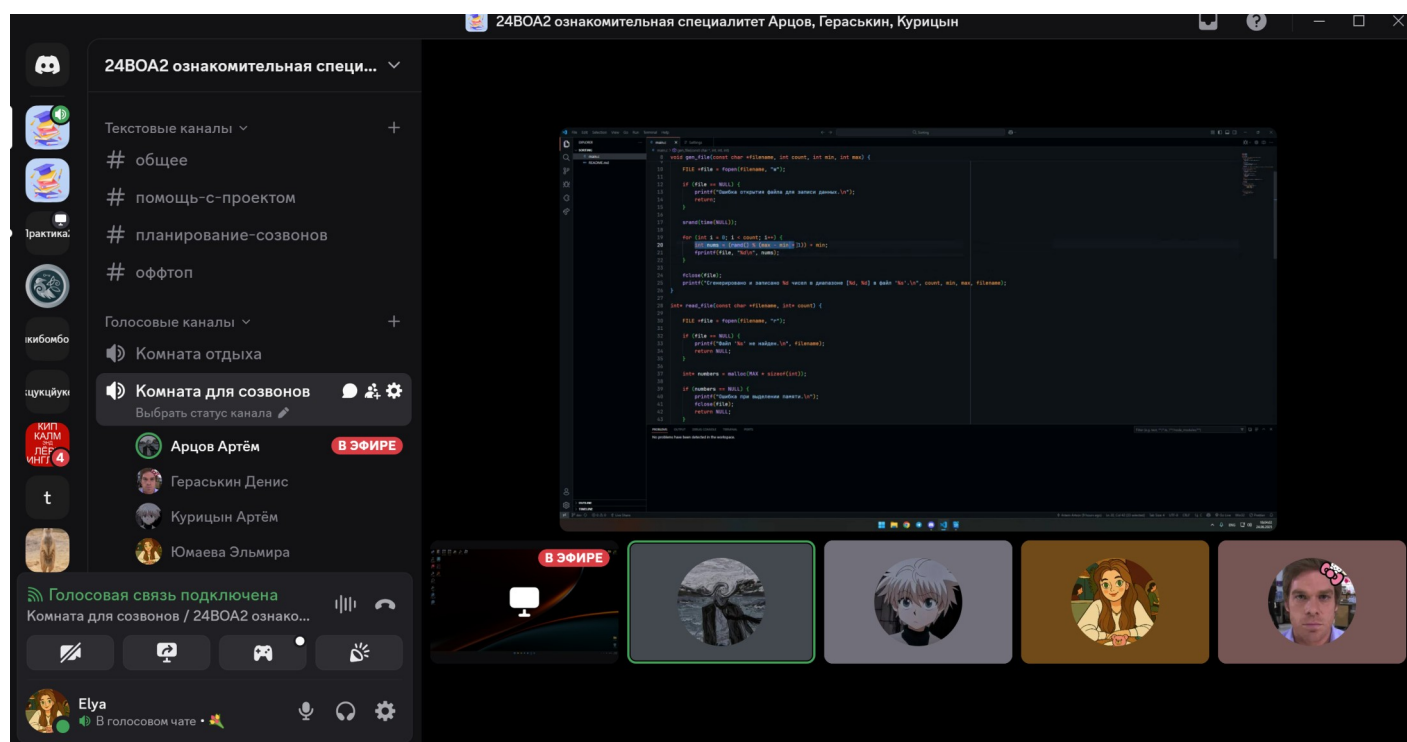


Рисунок 10 — Работа в *Discord*

Во время работы над данной практикой наша бригада осуществляла совместную работу в *GitHub*.

Мной было написано: функция сортировки методом пузырька и функции по работе с файлами, результат был загружен на удаленный репозиторий *Github*, сначала на ветку *dev*, затем после завершения работы над программой результаты были слиты в ветку *main*.

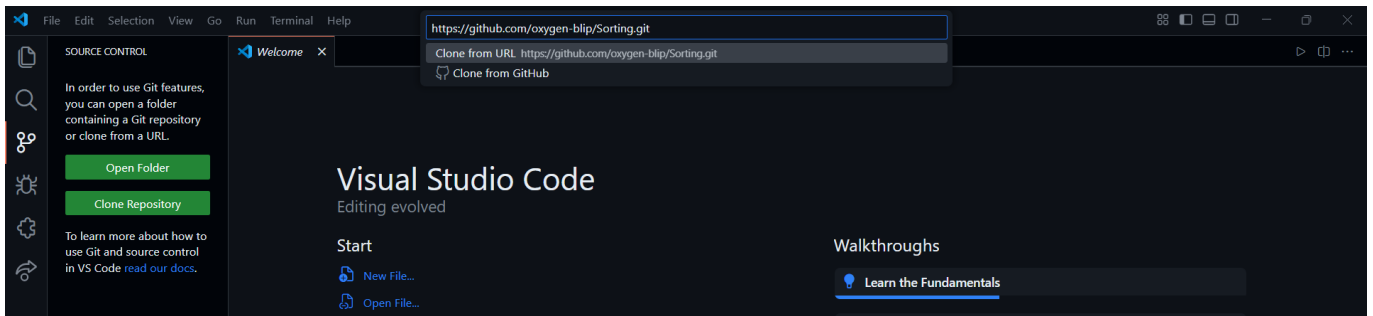


Рисунок 11 — Клонирование репозитория

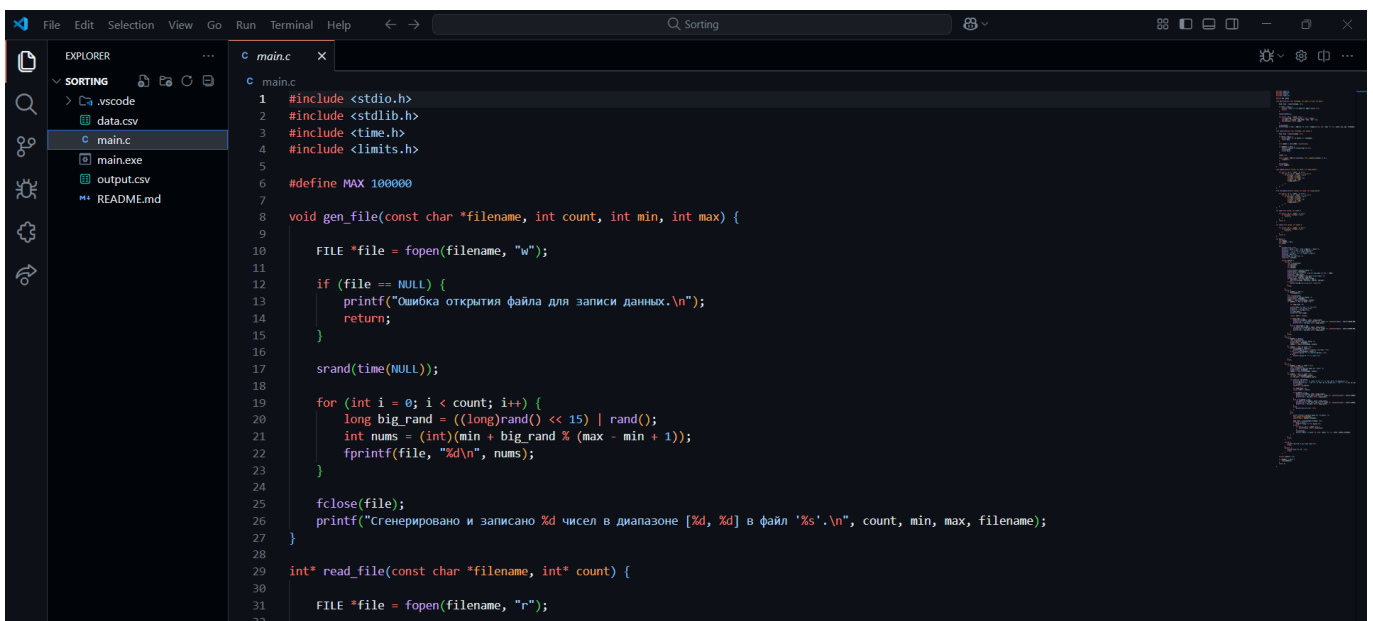


Рисунок 12 — Результат клонирования

```

commit b4efc6aaaa4c16747db4c556060d2b1c00f299a4
Author: Artem <alearc@mail.ru>
Date: Tue Jun 24 17:18:30 2025 +0300

    Update README.md

commit 8c29ca875370d7acb08f5c7b60aea5a626c803ad
Author: Artem Artsov <alearc@mail.ru>
Date: Tue Jun 24 14:13:32 2025 +0300

    README

commit be7fcb0844e47b02ce0b257ba548e5917ca5f30f
Author: Artem Artsov <alearc@mail.ru>
Date: Tue Jun 24 09:04:53 2025 +0300

    fix(bubble_sort)

commit 6e6d19b2ecdce46d06ae1cc85397b6276d3803ce
Author: Artem Artsov <alearc@mail.ru>
Date: Tue Jun 24 08:59:39 2025 +0300

    first part of the work

commit d930dcc7d04c52034fd263e3ab45e471ae9228ec
Author: Artem Artsov <alearc@mail.ru>
Date: Mon Jun 23 18:22:28 2025 +0300

    first commit
(END)

```

Рисунок 13 — Загрузки результатов в ветку *dev*

Ссылка на удаленный репозиторий:

<https://github.com/oxygen-blip/Sorting>

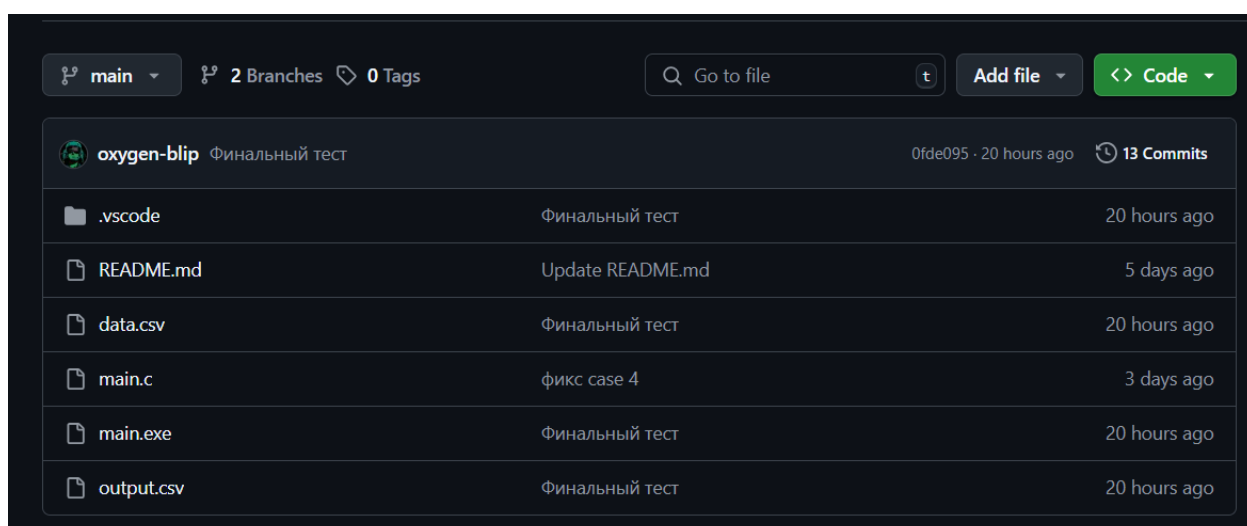


Рисунок 14 — Ветка *main*

Commits

main

All users

All time

Commits on Jun 28, 2025

Финальный тест

oxygen-blip committed 20 hours ago

0fde095

<>

Commits on Jun 26, 2025

+ данные

Klodinsky committed 2 days ago

cc6e356

<>

Merge pull request #1 from oxygen-blip/dev

oxygen-blip authored 2 days ago

Verified

84014c3

<>

+ функция, позволяющая сортировать числа от большего к меньшему;

Klodinsky committed 2 days ago

db2ac53

<>

фикс case 4

Klodinsky committed 3 days ago

0d04028

<>

+ функция, позволяющая сортировать числа от большего к меньшему;

Klodinsky committed 3 days ago

44bf88d

<>

Commits on Jun 25, 2025

fix(menu)

ArtemPGU committed 4 days ago

25fd47e

<>

Menu creation

ArtemPGU committed 4 days ago

ff988a9

<>

Commits on Jun 24, 2025

Update README.md

oxygen-blip authored 5 days ago

Verified

b4efc6a

<>

README

oxygen-blip committed 5 days ago

8c29ca8

<>

fix(bubble_sort)

oxygen-blip committed 5 days ago

be7fcb0

<>

first part of the work

oxygen-blip committed 5 days ago

6e6d19b

<>

Commits on Jun 23, 2025

first commit

oxygen-blip committed last week

d930dc

<>

Рисунок 15 — История коммитов

Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервисов *GitHub* и *WEEK*, навыки использования программы *Git Bash*. Был изучен алгоритм сортировки вставками.

Мной был написан алгоритм, который реализует сортировку методом пузырька, а также реализована работа с текстовыми файлами.

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

В дальнейшем программу можно улучшить путем подключения упрощающих и ускоряющих реализацию данной сортировки библиотек и улучшения графического интерфейса пользователя для более приятного пользования программой.

Список используемой литературы

1. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание.: Пер. с англ. – М., 2009.
2. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. - Введ. 1991-01-01. - М.: Стандартиформ, 2006. - 64 с.
3. Bubble Sort Algorithm [Электронный ресурс] // GeeksforGeeks. - URL: <https://www.geeksforgeeks.org/bubble-sort/> (дата обращения: 28.06.2025).

Приложение А. Листинг программы

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

#define MAX 100000

void gen_file(const char *filename, int count, int min, int max)
{
    FILE *file = fopen(filename, "w");

    if (file == NULL) {
        printf("Ошибка открытия файла для записи данных.\n");
        return;
    }

    srand(time(NULL));

    for (int i = 0; i < count; i++) {
        long big_rand = ((long)rand() << 15) | rand();
        int nums = (int)(min + big_rand % (max - min + 1));
        fprintf(file, "%d\n", nums);
    }

    fclose(file);
    printf("Сгенерировано и записано %d чисел в диапазоне [%d, %d] в файл '%s'.\n", count, min, max, filename);
}

int* read_file(const char *filename, int* count) {
    FILE *file = fopen(filename, "r");

    if (file == NULL) {
        printf("Файл '%s' не найден.\n", filename);
        return NULL;
    }

    int* numbers = malloc(MAX * sizeof(int));

    if (numbers == NULL) {
        printf("Ошибка при выделении памяти.\n");
        fclose(file);
        return NULL;
    }
}
```

```

    }

    *count = 0;

    while (*count < MAX && fscanf(file, "%d", &numbers[*count])
== 1) {
        (*count)++;
    }

    fclose(file);
    return numbers;
}

void bubble_sort(int *array, int count, int *swap_count) {

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
                (*swap_count)++;
            }
        }
    }
}

void rev_bubble_sort(int *array, int count, int *swap_count){

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            if (array[j] < array[j + 1]) {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
                (*swap_count)++;
            }
        }
    }
}

int check (int *array, int count) {

    for (int i = 0; i < count - 1; i++) {
        if (array[i] > array[i + 1]) {
            return 0;
        }
    }
    return 1;
}

```

```

int check1 (int *array, int count) {

    for (int i = 0; i < count - 1; i++) {
        if (array[i] < array[i + 1]) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int choice;
    int* numbers = NULL;
    int count;

    do{
        printf("\nМеню:\n");
        printf("1. Сгенерировать числа и записать в файл\n");
        printf("2. Отсортировать числа из файла\n");
        printf("3. Проверить сортировку чисел\n");
        printf("4. Сохранить отсортированные числа\n");
        printf("0. Выход\n");
        printf("Выберите действие: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: {
                char filename[256];
                int num_count;
                int num_min;
                int num_max;

                printf("Введите название файла: ");
                scanf("%255s", &filename);
                printf("Введите количество чисел для генерации (до
%d): ", MAX);
                scanf("%d", &num_count);
                printf("Введите диапазон генерации(через пробел):
");
                scanf("%d %d", &num_min, &num_max);
                if (num_count > 0 && num_count <= MAX) {
                    gen_file(filename, num_count, num_min,
num_max);
                } else {
                    printf("Недопустимое количество чисел!\n");
                }
                break;
            }
            case 2: {
                if (numbers != NULL) {

```



```

        free(numbers);
    }
    char filename[256];
    printf("Введите название файла: ");
    scanf("%255s", filename);
    numbers = read_file(filename, &count);
    if (numbers != NULL && count > 0) {

        int swap_count = 0;

        printf("Выберите тип сортировки:\n");
        printf("1. По возрастанию\n");
        printf("2. По убыванию\n");
        int sort_type;
        scanf("%d", &sort_type);

        clock_t start = clock();

        if (sort_type == 1){
            bubble_sort(numbers, count,
&swap_count);
            printf("\nСортировка завершена за %.4f
секунд.\n", (double)(clock() - start)/CLOCKS_PER_SEC);
            printf("Кол-во операций: %d\n",
swap_count);
        }
        else if (sort_type == 2){
            rev_bubble_sort(numbers, count,
&swap_count);
            printf("\nСортировка завершена за %.4f
секунд.\n", (double)(clock() - start)/CLOCKS_PER_SEC);
            printf("Кол-во операций: %d\n",
swap_count);
        }
    }
    break;
}
case 3: {
    if (numbers == NULL) {
        char filename[256];
        printf("Введите название файла: ");
        scanf("%255s", filename);
        numbers = read_file(filename, &count);
    }
    if (numbers != NULL && count > 0) {
        if (check(numbers, count)) {
            printf("\nЧисла отсортированы по
возрастанию.\n");
        } else if (check1(numbers, count)) {
            printf("\nЧисла отсортированы по
убыванию.\n");
        } else{

```

```

        printf("\nЧисла не отсортированы.\n");
    }
}
break;

}
case 4: {
    if (numbers == NULL || count <= 0) {
        char filename[256];
        printf ("Введите название файла для чтения:
");

        scanf("%255s", filename);
        numbers = read_file(filename, &count);
    }
    if (numbers != NULL && count > 0) {
        int sort = check(numbers, count);
        int rev_sort = check1(numbers, count);

        if (!sort && !rev_sort){
            printf("Числа не отсортированы! Хотите
отсортировать перед сохранением?\n ");
            printf("Введите:\n1 - чтобы
отсортировать по возрастанию\n2 - чтобы отсортировать по
убыванию\nВаш выбор:");

            int s_choice;
            scanf("%d", &s_choice);

            int swap_count = 0;
            clock_t start = clock();

            if (s_choice == 1){
                bubble_sort(numbers, count,
&swap_count);

                printf("\nСортировка завершена за
%.4f секунд.\n", (double)(clock() - start)/CLOCKS_PER_SEC);
                printf("Кол-во операций: %d\n",
swap_count);
            }
            else if (s_choice == 2){
                rev_bubble_sort(numbers, count,
&swap_count);

                printf("\nСортировка завершена за
%.4f секунд.\n", (double)(clock() - start)/CLOCKS_PER_SEC);
                printf("Кол-во операций: %d\n",
swap_count);
            }
            else{
                printf("Неверный выбор.\n");
            }
        }
    }
    else
    {

```

```

сохранения: ");
printf ("Введите название файла для
char output_filename[256];
scanf("%255s", &output_filename);

FILE *file = fopen(output_filename,
"w");

if (file == NULL) {
    printf("Ошибка открытия файла!\n");
} else {
    for (int i = 0; i < count; i++) {
        fprintf(file, "%d\n",
numbers[i]);
    }
    fclose(file);
    printf("Успешно сохранено %d чисел
в файле '%s'.\n", count, output_filename);
}
}
break;
}

case 0: {
    printf("Работа программы завершена.\n");
    break;
}
default: {
    printf("Неверный выбор.\n");
    break;
}
}

} while (choice != 0);

if (numbers != NULL) {
    free(numbers);
}
return 0;

}

```