

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
(2022/2023 учебный год)

_____ Зубриядова Арина Аскольдывна

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения _____ 1 _____ семестр _____ 2 _____

Период прохождения практики с 29.06.2023 по 12.07.2023

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**

(2022/2023 учебный год)

_____ Зубриядова Арина Аскольдывна

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения _____ 1 _____ семестр _____ 2 _____

Период прохождения практики с 29.06.2023 по 12.07.2023

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	29.06.2023 - 29.06.2023	
2	Подбор и изучение материала по теме работы	15	30.06.2023 – 02.07.23	
3	Разработка алгоритма	43	02.07.23 – 06.07.23	
4	Описание алгоритма и программы	18	6.07.23 – 08.07.23	
5	Тестирование	5	08.07.23 – 08.07.23	
6	Получение и анализ результатов	10	08.07.23 – 10.07.23	
7	Оформление отчёта	15	10.07.23 – 12.07.2023	
	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

(2022/2023 учебный год)

Зубриядова Арина Аскольдодна

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Зубриядова А.А. выполняла практическое задание «Сортировка вставками». На первоначальном этапе были изучен и проанализирован алгоритм сортировки вставками, был выбран метод решения и язык программирования С, на котором была написана программа сортировки массива методом вставок. Также, осуществила подсчёт времени выполнения сортировки и количества перестановок. Протестировала и отладила программу. Оформила отчёт.

Бакалавр Зубриядова А.А. " " 2023 г.

Руководитель Зинкин С.А. " " 2023 г.
практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЗЫВ

О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2022/2023 учебный год)

Зубриядова Арина Аскольдодна

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения практики Зубриядова А.А. решала следующие задачи: изучение алгоритма сортировки вставками, анализ работы алгоритма, сравнение существующих методов сортировки, произведение подсчёта времени выполнения сортировки и количества перестановок, тестирование и отладка программы.

За период выполнения практики были освоены основные понятия и технологии сортировки вставками. Во время выполнения работы Зубриядова А.А. показала 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 Зубриядова А.А. заслуживает оценки « ».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2023 г.

Содержание

Введение	2
1 Постановка задачи.....	3
1.1 Достоинства алгоритма сортировки вставками	3
1.2 Недостатки алгоритма сортировки вставками.....	3
1.3 Типичные сценарии применения данного алгоритма	3
2 Выбор решения	4
3 Описание программы.....	5
4 Схемы программы.....	6
4.1 Блок-схема программы	6
5 Тестирование программы	7
5.1 Тестирование на разных наборах данных.....	7
5.2 Анализ полученных результатов тестирования (анализ работы алгоритма)	7
6 Отладка	9
7 Совместная разработка	10
Заключение.....	14
Список используемой литературы	15
Приложение А	16
Приложение Б Листинг	19

Введение

Сортировка данных на сегодняшний день при современном развитии компьютерных технологий является одним из наиболее распространенных процессов современной обработки данных. Задачи на сортировку данных встречаются очень часто в различных профессиональных сферах деятельности.

Алгоритмы сортировки очень широко распространяются практически во всех задачах обработки информации. Они образуют отдельный класс алгоритмов, применяются с целью осуществления последующего более быстрого поиска.

Важность сортировки основана на том факте, что на ее примере можно показать многие основные фундаментальные приемы и методы построения алгоритмов. Сортировка является хорошим примером огромного разнообразия алгоритмов, которые выполняют одну и ту же задачу. Кроме того, многие из них имеют определенные преимущества друг перед другом. За счет усложнения алгоритма можно добиться существенного увеличения эффективности и быстродействия алгоритма по сравнению с более простыми методами. Как правило, термин сортировка понимают, как процесс перестановки объектов некоторого множества в определенном порядке.

Сортировка вставками является стабильной. Алгоритм не самый лучший с точки зрения производительности, но традиционно более эффективен, чем большинство других простых алгоритмы, такие как сортировка выбором или же пузырьковая сортировка. Сортировка вставками также используется в гибридной сортировке, которая сочетает в себе различные алгоритмы для повышения производительности.

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 вставками над данными, находящимися в массиве, записать отсортированные данные в другой файл, 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 вставками

- алгоритм удобен для работы с массивами небольшого размера или на почти отсортированном наборе данных;
- алгоритм эффективен при работе со списками;
- простая реализация алгоритма.

1.2 Недостатки алгоритма сортировки вставками

- очень много перемещений элементов массива;
- высокая алгоритмическая сложность $O(n^2)$;
- не рекомендуется для сортировки больших массивов.

1.3 Типичные сценарии применения данного алгоритма

- товары в магазине (сортировка по цене, году выпуска, габаритам, весу, срокам поставки);
- студенты в вузе (сортировка по среднему балу, кол-ву прогулов, уровню IQ, числу хвостов, ФИО);
- города/страны (сортировка по населению, рождаемости, ВВП, ВВП на душу населения);
- астрономические объекты (масса, размеры, плотность).

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. Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Для удобства совместной разработки был использован сервис WEEK. 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 Week.

3 Описание программы

При запуске программы выводится меню из пяти пунктов:

- а) сортировка случайных значений по возрастанию;
- б) сортировка случайных значений по убыванию;
- в) сортировка возрастающих значений по убыванию;
- г) сортировка возрастающих значений по возрастанию;
- д) esc – выход.

Пользователю требуется выбрать тот пункт, который ему требуется. При выборе при выборе пунктов под буквами а-г выводится сообщение, в котором пользователю необходимо ввести количество значений для сортировки.

После того, как данные были введены, генерируется массив из случайных чисел, эти числа записываются в файл input.txt.

Далее над этими данными выполняется сортировка вставками, при которой массив постепенно перебирается слева направо. При этом элемент сравнивается со всеми предыдущими элементами и размещается так, чтобы оказаться в подходящем месте среди ранее упорядоченных элементов. Так происходит до тех пор, пока набор входных данных не будет исчерпан.

После этого отсортированный массив записывается в файл output.txt.

Программа так же осуществляет подсчет количества перестановок элементов массива и времени, которое заняла сортировка.

При выборе пункта меню под буквой д программа завершает выполнение.

Подробный алгоритм работы программы и функции сортировки представлен в подразделе 4.1 на рисунке 1.

Листинг программы приведен в приложении Б.

4 Схемы программы

4.1 Блок-схема программы

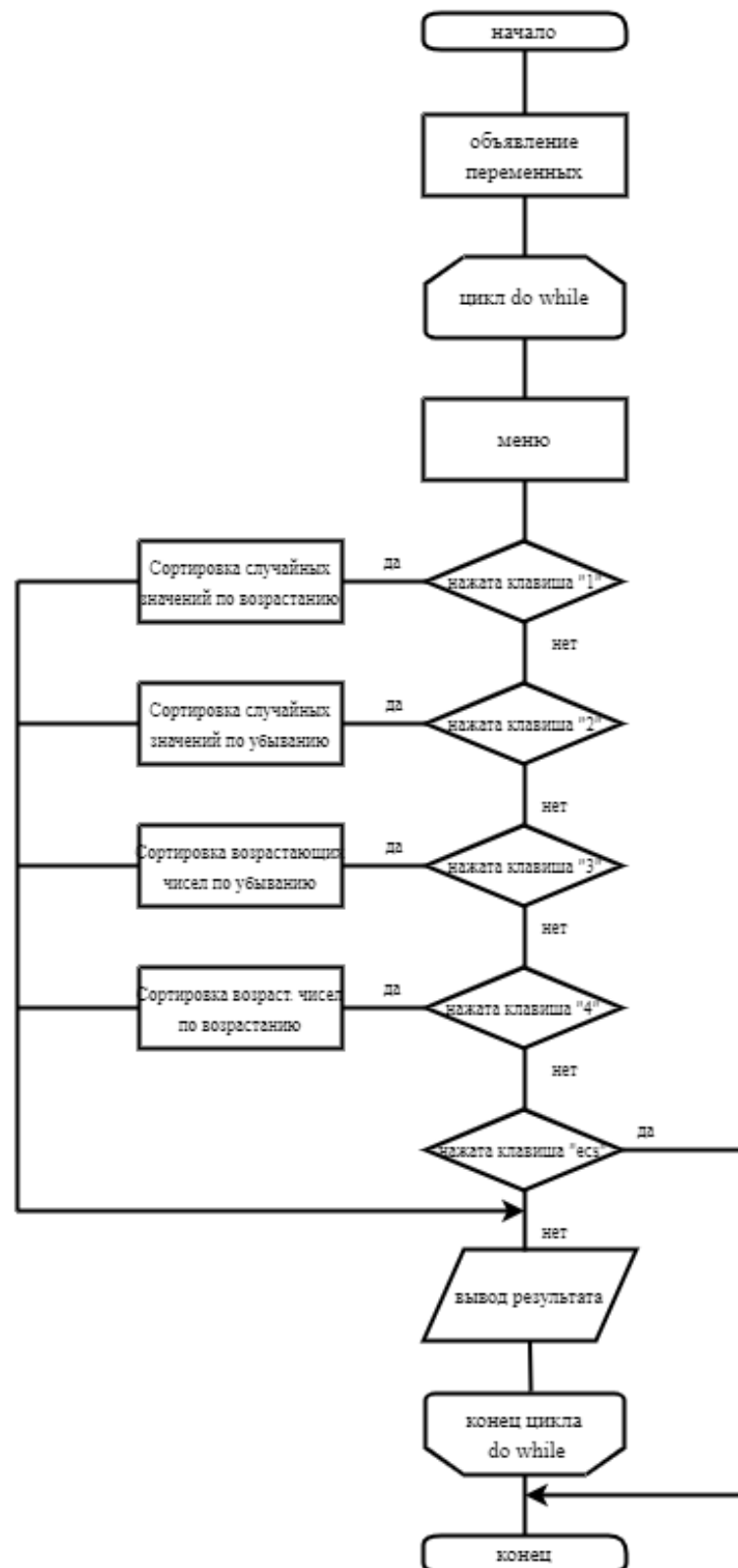


Рисунок 1 – Блок-схема программы

5 Тестирование программы

5.1 Тестирование на разных наборах данных

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в Приложении А на рисунках А.1 - А.11.

Таблица 1 – Тестовый набор данных

	Размер массива size	Время выполнения сортировки в секундах	Количество перестановок
1	10000	0.130	24939558
2	20000	0.526	100308281
3	30000	1.180	226460657
4	40000	2.080	399509966
5	50000	3.259	625919110
6	60000	4.717	901087418
7	70000	6.366	1218916785
8	80000	8.277	1599860736
9	90000	9.875	2029707626
10	100000	11.635	22472407232
11	110000	13.615	26435510235

5.2 Анализ полученных результатов тестирования (анализ работы алгоритма)

На основании анализа данных, полученных в результате тестирования алгоритма сортировки вставками, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается линейно, то есть с увеличением количества элементов пропорционально увеличивается время работы программы.



Рисунок 2 – Результаты тестирования

6 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 останова 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 и устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Был использован метод бинарного поиска, он включает в себя разделение частей кода для упрощения процесса отладки. Это может быть особенно полезно, если причина ошибки находится в начале языка программирования, а фактическая ошибка ближе к концу.

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.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. После завершения написания программы, мною были выявлены и исправлены ошибки.

7 Совместная разработка

Для удобства совместной разработки был использован сервис WEEK.

Определили задачи проекта, назначили приоритет задачам.

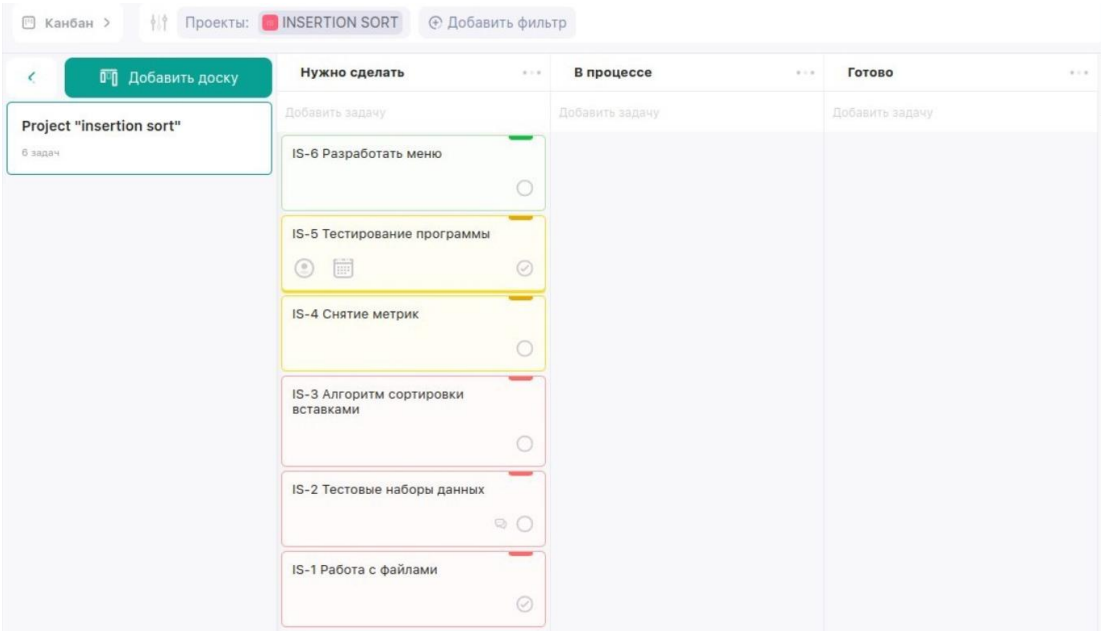


Рисунок 3 – Определение задач проекта

Разделили роли, назначили исполнителей задачам.

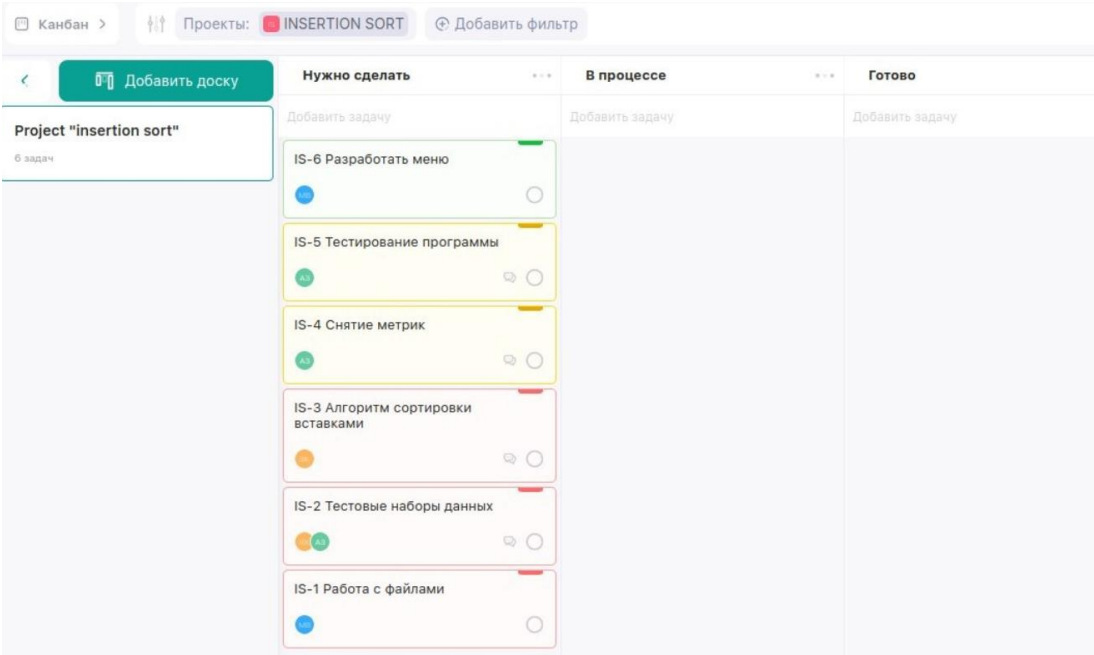


Рисунок 4 – Распределение задач проекта

Обсуждали выполнение задачи на канбан-доске.

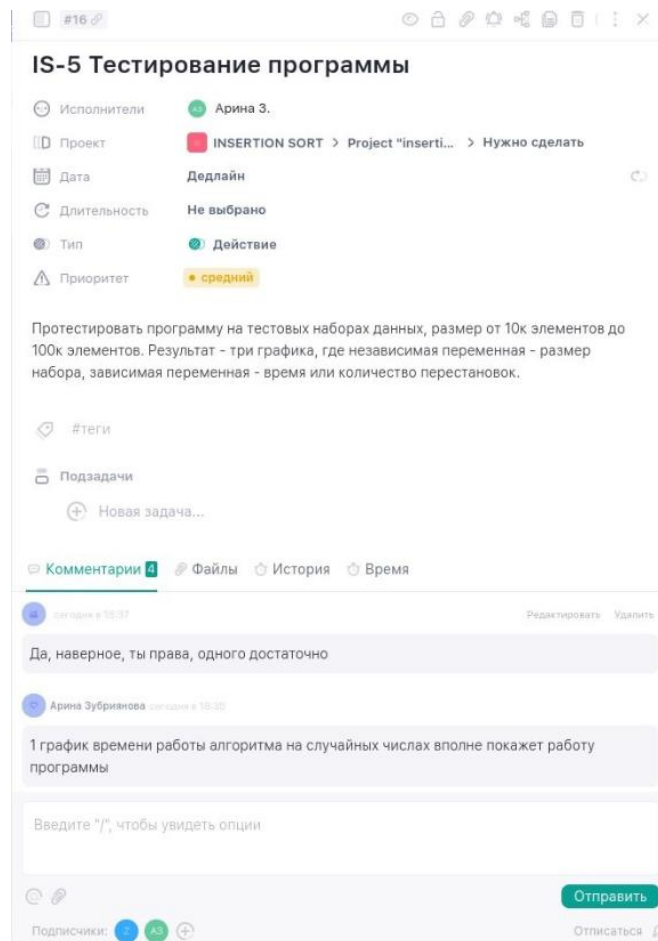


Рисунок 5 – Обсуждение задач проекта

Корректировали статус задач по мере выполнения.

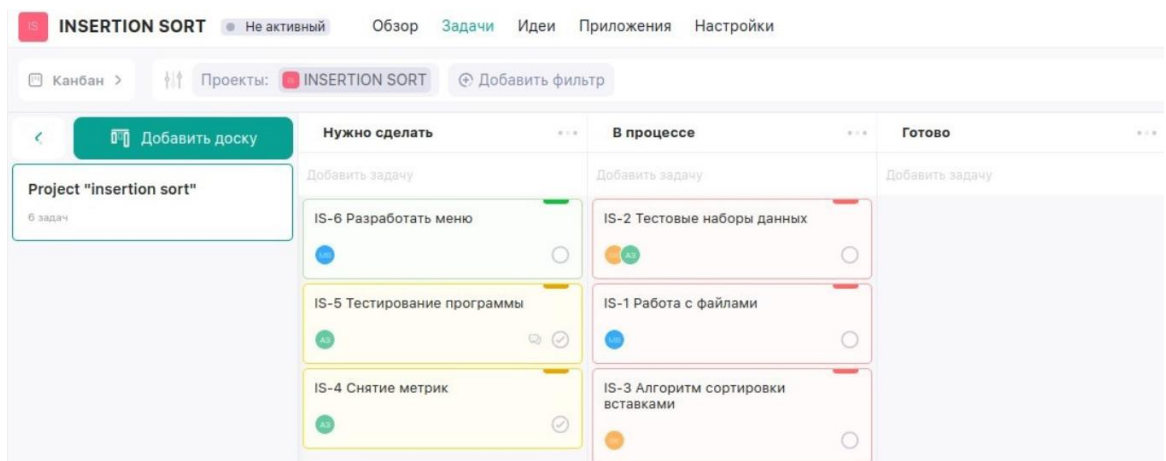


Рисунок 6 – Корректирование задач

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

Данная программа была загружена на компьютер, с помощью git clone <ссылка>.

```
MINGW64:/d/Pr/Praktika
Арина@DESKTOP-AD70TMC MINGW64 /d/Pr (master)
$ git init
Reinitialized existing Git repository in D:/Pr/.git/

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr (master)
$ git clone https://github.com/Margarita07460/Praktika.git
Cloning into 'Praktika'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 29 (delta 5), reused 26 (delta 5), pack-reused 0
Receiving objects: 100% (29/29), 299.29 KiB | 526.00 KiB/s, done.
Resolving deltas: 100% (5/5), done.
```

Рисунок 7 – Загрузка папки с удаленного репозитория

Мной был добавлен алгоритм, считающий количество перестановок и время работы программы в секундах.

Я загрузила обновленный код программы на удаленный репозиторий GitHub.

```
Арина@DESKTOP-AD70TMC MINGW64 /d/Pr (master)
$ cd Praktika

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git remote -v
origin https://github.com/Margarita07460/Praktika.git (fetch)
origin https://github.com/Margarita07460/Praktika.git (push)

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git add --all

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git commit -m "время сортировки и количество перестановок"
[main 8c8e647] время сортировки и количество перестановок
21 files changed, 69 insertions(+), 9 deletions(-)
create mode 100644 .vs/praktish1/project-colors.json
create mode 100644 .vs/praktish1/v17/.suo
create mode 100644 .vs/praktish1/v17/Browse.VC.db
create mode 100644 .vs/praktish1/v17/ipch/AutoPCH/d4b128d9f10c6c37/SOURCE.ipch

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git log
commit 8c8e647462f2547bd54848dc57a022f0ed53b7d4 (HEAD -> main)
Author: Arina <zubr230305@gmail.com>
Date: Sat Jul 1 23:53:59 2023 +0300

    время сортировки и количество перестановок

commit f1d53af8db8b11b60e7d7d73d0b1b57e8902e8ca (origin/main, origin/HEAD)
Author: Vika <vikka_kondrateva@mail.ru>
Date: Sat Jul 1 20:11:56 2023 +0300

    алгоритм

commit ef17bbc961b044e2158b61a5aec19ce395dbf7ab
Author: Margarita07460 <132387176+Margarita07460@users.noreply.github.com>
Date: Sat Jul 1 19:18:57 2023 +0300

    Initial commit

Арина@DESKTOP-AD70TMC MINGW64 /d/Pr/Praktika (main)
$ git push
Enumerating objects: 52, done.
Counting objects: 100% (52/52), done.
Delta compression using up to 6 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (32/32), 1.28 MiB | 279.00 KiB/s, done.
Total 32 (delta 15), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (15/15), completed with 14 local objects.
To https://github.com/Margarita07460/Praktika.git
f1d53af..8c8e647 main -> main
```

Рисунок 8 – Загрузка измененной программы на удаленный репозиторий

Ссылка на удаленный репозиторий:

<https://github.com/Margarita07460/Praktika.git>

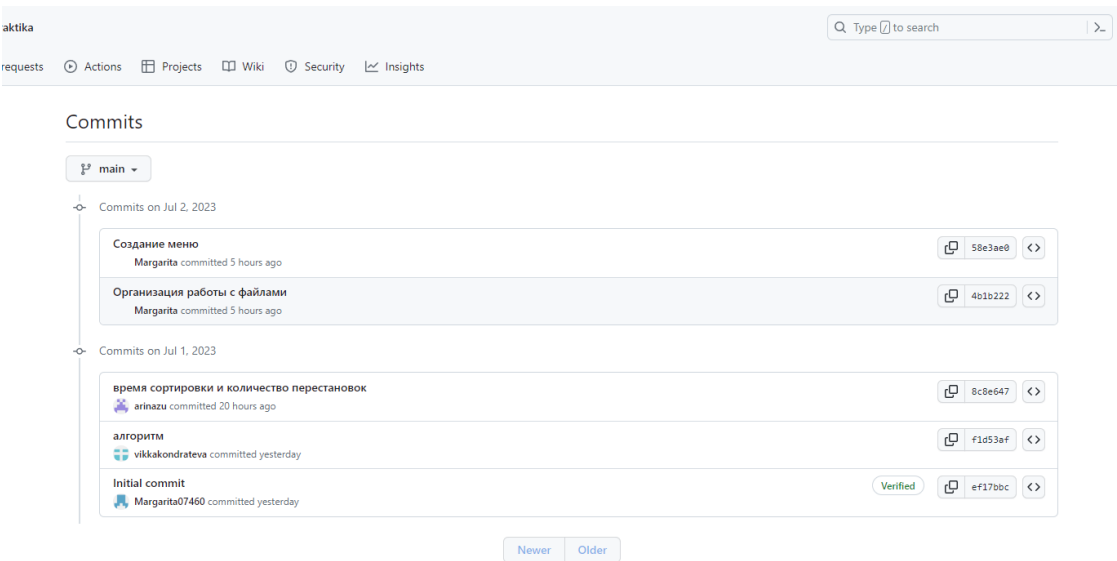


Рисунок 9 – История коммитов ветки main

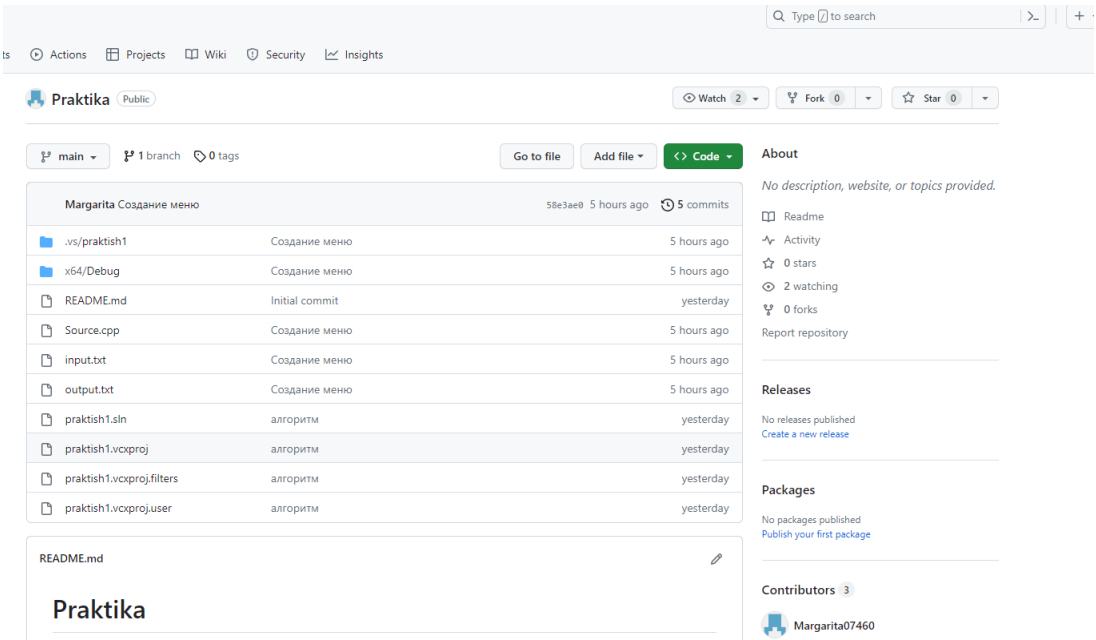


Рисунок 10 – Ветка main

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.

Мною был написан алгоритм, считающий количество перестановок элементов массива во время сортировки и время работы программы в секундах. Было выполнено тестирование программы на разных наборах данных и отладка данной программы.

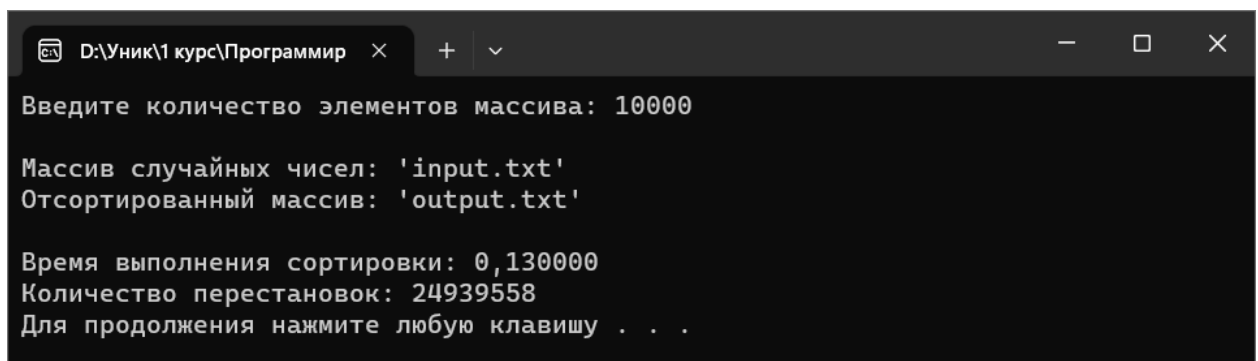
При выполнении практической работы были улучшены базовые навыки программирования на языке C. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

В дальнейшем программу можно улучшить путем подключения упрощающих реализацию данной сортировки библиотек и улучшения графического интерфейса.

Список используемой литературы

1. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.
2. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание.: Пер. с англ. – М., 2009.
3. Сортировка вставками [Электронный ресурс] – URL: <https://ru.wikipedia.org> (дата обращения: 02.07.2023 г)

Приложение А. Результаты тестирования программы

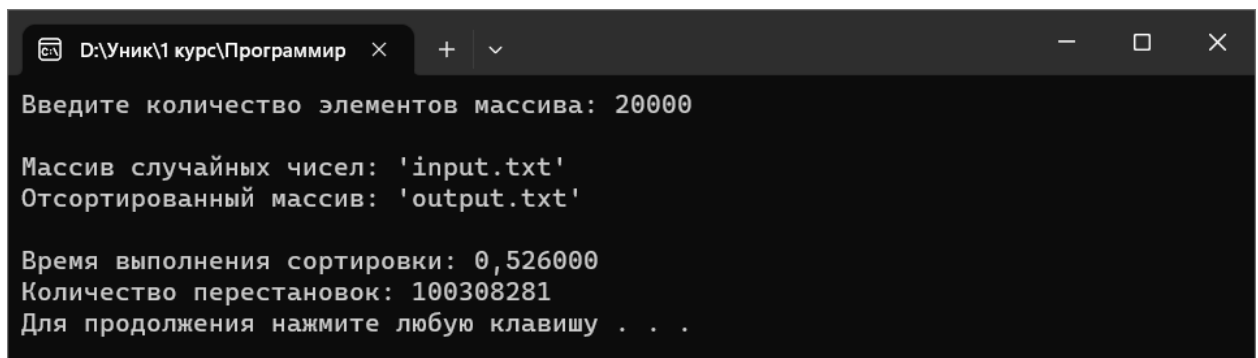


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 10000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 0,130000
Количество перестановок: 24939558
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.1

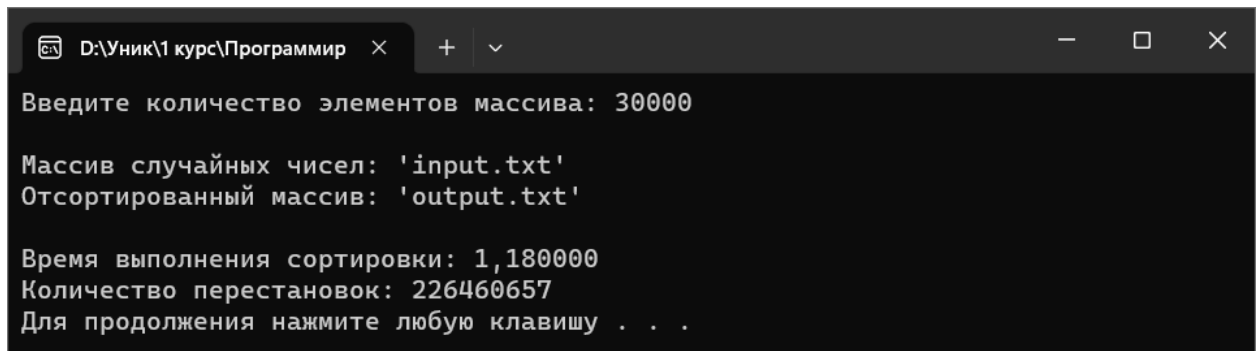


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 20000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 0,526000
Количество перестановок: 100308281
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.2

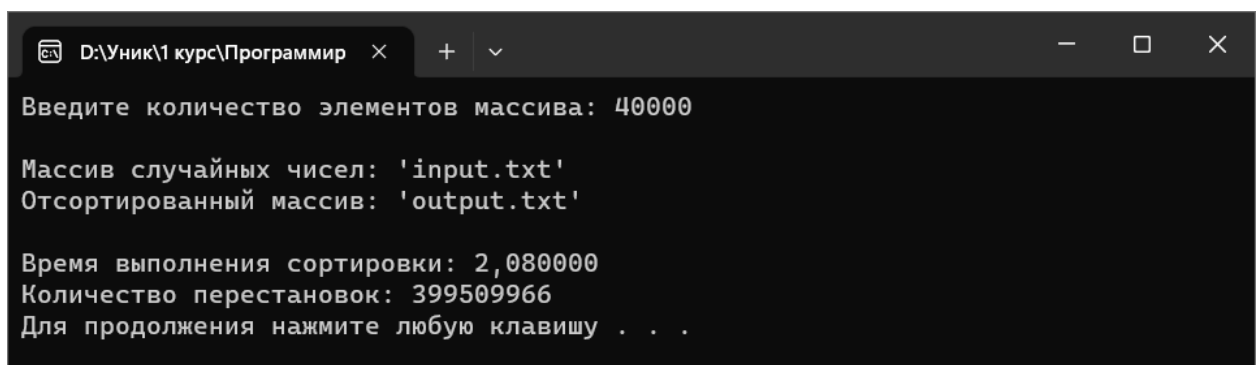


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 30000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 1,180000
Количество перестановок: 226460657
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.3

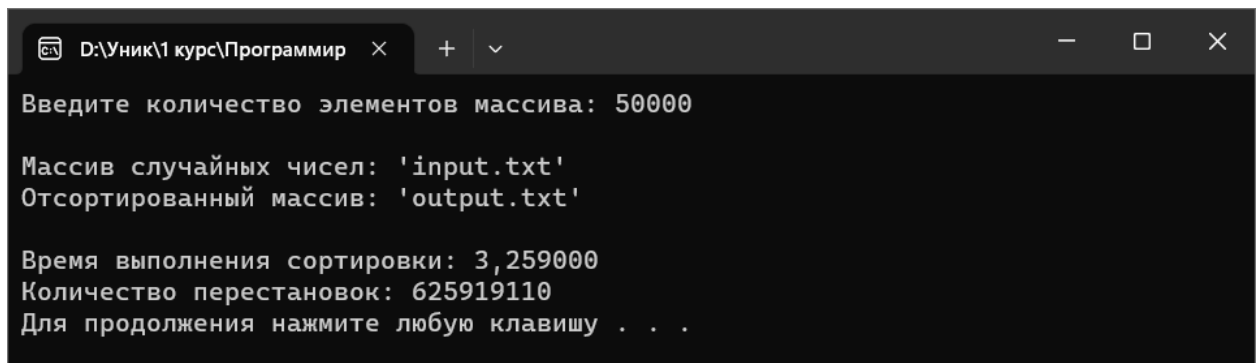


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 40000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 2,080000
Количество перестановок: 399509966
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.4

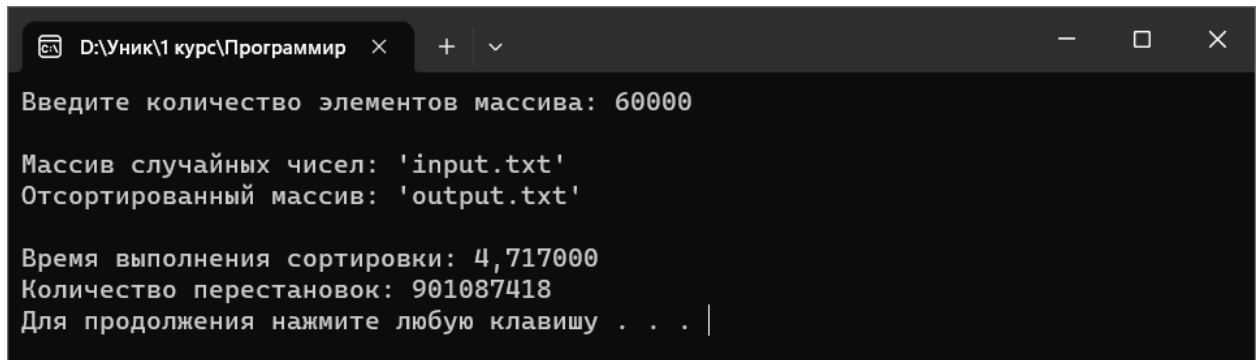


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 50000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 3,259000
Количество перестановок: 625919110
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.5

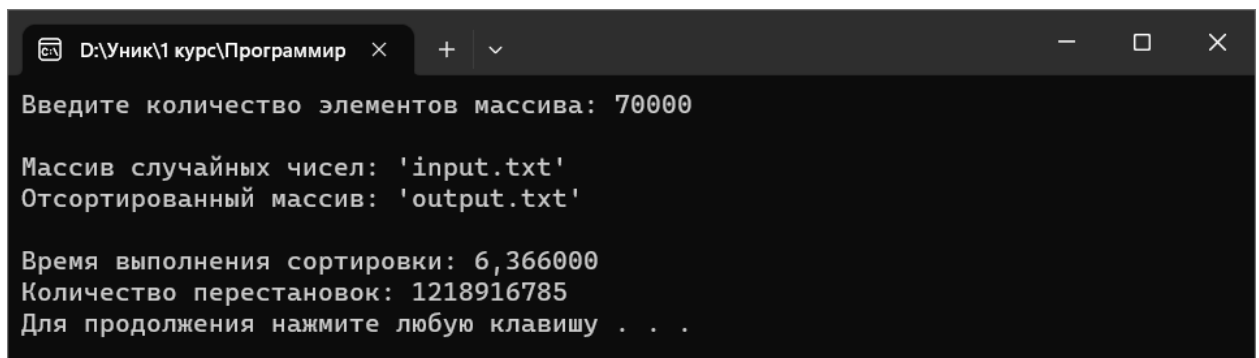


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 60000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 4,717000
Количество перестановок: 901087418
Для продолжения нажмите любую клавишу . . . |
```

Рисунок А.6

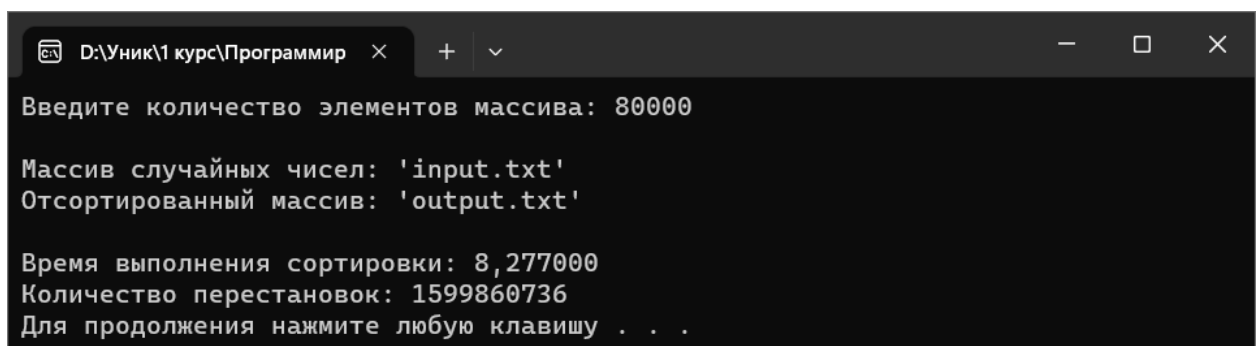


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 70000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 6,366000
Количество перестановок: 1218916785
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.7

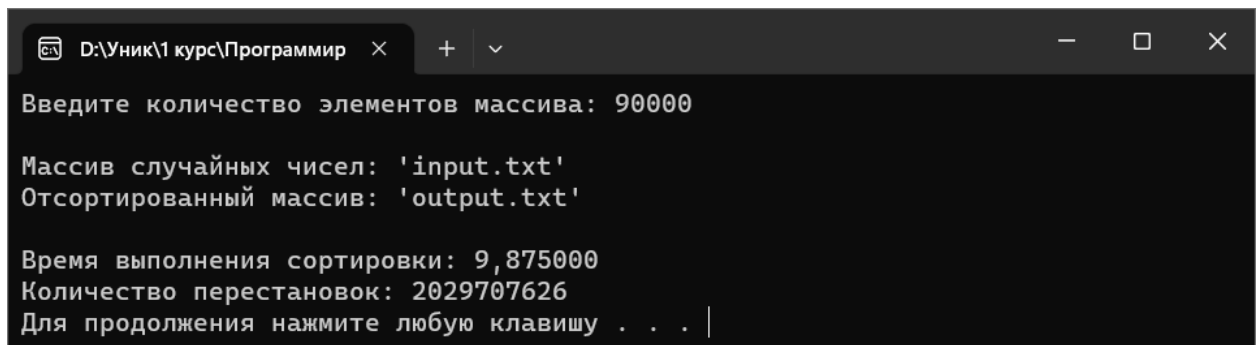


```
D:\Уник\1 курс\Программир x + v - □ x
Введите количество элементов массива: 80000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 8,277000
Количество перестановок: 1599860736
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.8

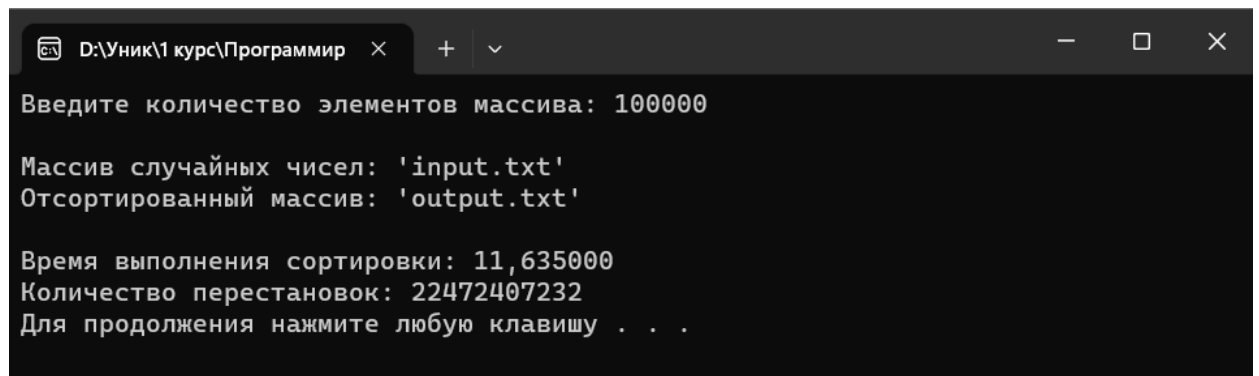


```
D:\Уник\1 курс\Программир >
Введите количество элементов массива: 90000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 9,875000
Количество перестановок: 2029707626
Для продолжения нажмите любую клавишу . . . |
```

Рисунок А.9

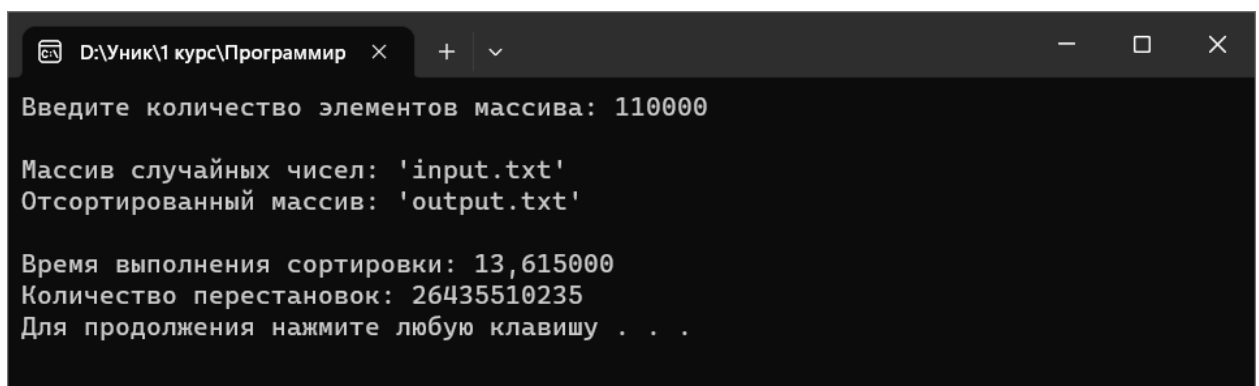


```
D:\Уник\1 курс\Программир >
Введите количество элементов массива: 100000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 11,635000
Количество перестановок: 22472407232
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.10



```
D:\Уник\1 курс\Программир >
Введите количество элементов массива: 110000

Массив случайных чисел: 'input.txt'
Отсортированный массив: 'output.txt'

Время выполнения сортировки: 13,615000
Количество перестановок: 26435510235
Для продолжения нажмите любую клавишу . . .
```

Рисунок А.11

Приложение Б. Листинг программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>

int main()
{
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));

    FILE* f;
    int m, size;
    int* array;
    char ch;

    do {
        system("cls");
        printf("МЕНЮ:\n");
        printf("1 - Сортировка случайных значений по возрастанию\n");
        printf("2 - Сортировка случайных значений по убыванию\n");
        printf("3 - Сортировка возрастающих значений по убыванию\n");
        printf("4 - Сортировка возрастающих значений по возрастанию\n");

        printf("\necs - выход\n");
        ch = _getch();

        switch (ch) {
            case '1':
                //Сортировка случайных значений по возрастанию
                system("cls");
```



```

printf("Введите количество элементов массива: ");
scanf("%d", &size);
printf("\n");
array = (int*)malloc(size * sizeof(int));

printf("Массив случайных чисел: 'input.txt'\n");
f = fopen("input.txt", "w");
for (int i = 0; i < size; i++)
{
    array[i] = rand() - rand();
    fprintf(f, "%d ", array[i]);
}
fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time_t start = clock(); //время до сортировки

long count = 0;
for (int i = 1; i < size; i++)
{
    m = array[i];
    for (int j = i - 1; j >= 0; j--)
    {
        if (array[j] < m)
            break;
        array[j + 1] = array[j];
        array[j] = m;
        count++;
    }
}

```

```

        time_t stop = clock();                //время после сортировки
        for (int i = 0; i < size; i++) {
            fprintf(f, "%d ", array[i]);
        }
        fclose(f);
        double time = (stop - start) / 1000.0;    //время сортировки
        printf("\n");
        printf("Время выполнения сортировки: ");
        printf("%lf\n", time);
        printf("Количество перестановок: ");
        printf("%o\n", count);

        system("pause");
        break;
    }

    switch (ch) {
        case '2':
            //Сортировка случайных значений по убыванию
            system("cls");
            printf("Введите количество элементов массива: ");
            scanf("%d", &size);
            printf("\n");
            array = (int*)malloc(size * sizeof(int));

            printf("Массив случайных чисел: 'input.txt'\n");
            f = fopen("input.txt", "w");
            for (int i = 0; i < size; i++)
            {
                array[i] = rand() - rand();
                fprintf(f, "%d ", array[i]);
            }
            fclose(f);

```

```

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time_t start = clock(); //время до сортировки

long count = 0;
for (int i = 1; i < size; i++)
{
    m = array[i];
    for (int j = i - 1; j >= 0; j--)
    {
        if (array[j] > m)
            break;
        array[j + 1] = array[j];
        array[j] = m;
        count++;
    }
}
time_t stop = clock(); //время после сортировки
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", array[i]);
}
fclose(f);
double time = (stop - start) / 1000.0; //время сортировки
printf("\n");
printf("Время выполнения сортировки: ");
printf("%lf\n", time);
printf("Количество перестановок: ");
printf("%o\n", count);

system("pause");
break;

```

```

}

switch (ch) {
case '3':
//Сортировка возрастающих значений по убыванию
    system("cls");
    printf("Введите количество элементов массива: ");
    scanf("%d", &size);
    printf("\n");
    array = (int*)malloc(size * sizeof(int));

    printf("Массив случайных чисел: 'input.txt'\n");
    array[0] = rand();
    f = fopen("input.txt", "w");
    for (int i = 0; i < size; i++)
    {
        array[i + 1] = array[i] + rand() % 100 + 100;
        fprintf(f, "%d ", array[i]);

    }
    fclose(f);

    printf("Отсортированный массив: 'output.txt'\n");

    f = fopen("output.txt", "w");

    time_t start = clock(); //время до сортировки

    long count = 0;
    for (int i = 1; i < size; i++)
    {
        m = array[i];
        for (int j = i - 1; j >= 0; j--)
        {

```

```

        if (array[j] > m)
            break;

        array[j + 1] = array[j];
        array[j] = m;
        count++;
    }
}

time_t stop = clock();           //время после сортировки
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", array[i]);
}
fclose(f);

double time = (stop - start) / 1000.0;    //время сортировки
printf("\n");
printf("Время выполнения сортировки: ");
printf("%lf\n", time);
printf("Количество перестановок: ");
printf("%o\n", count);

system("pause");
break;
}

switch (ch) {
case '4':
//Сортировка возрастающих значений по возрастанию
    system("cls");
    printf("Введите количество элементов массива: ");
    scanf("%d", &size);
    printf("\n");
    array = (int*)malloc(size * sizeof(int));

    printf("Массив случайных чисел: 'input.txt'\n");

```

```

array[0] = rand();
f = fopen("input.txt", "w");
for (int i = 0; i < size; i++)
{
    array[i + 1] = array[i] + rand() % 100 + 100;
    fprintf(f, "%d ", array[i]);
}
fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time_t start = clock(); //время до сортировки

long count = 0;
for (int i = 1; i < size; i++)
{
    m = array[i];
    for (int j = i - 1; j >= 0; j--)
    {
        if (array[j] < m)
            break;
        array[j + 1] = array[j];
        array[j] = m;
        count++;
    }
}

time_t stop = clock(); //время после сортировки
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", array[i]);
}

```

```

fclose(f);
double time = (stop - start) / 1000.0;    //время сортировки
printf("\n");
printf("Время выполнения сортировки: ");
printf("%lf\n", time);
printf("Количество перестановок: ");
printf("%o\n", count);

system("pause");
break;
}
} while (ch != 27);
}

```