

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)

Кондратьева Виктория Игоревна

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)

Кондратьева Виктория Игоревна

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)

№ 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)

Кондратьева Виктория Игоревна

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»

Кондратьева В.И. выполняла практическое задание «Сортировка вставками». На первоначальном этапе были изучен и проанализирован алгоритм сортировки вставками, был выбран метод решения и язык программирования C, на котором была написана программа сортировки массива методом вставок. Отладил программу. Оформил отчёт.

Бакалавр Кондратьева В.И. _____ "___" _____ 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)

Кондратьева Виктория Игоревна

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 Кондратьева В.И. решала следующие задачи: создание алгоритма пузырьковой сортировки, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии сортировки вставками. Во время выполнения работы Кондратьева В.И. 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 Кондратьева В.И. заслуживает оценки «_____».

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

Содержание

Введение	2
1 Постановка задачи.....	3
2 Выбор решения.....	4
3 Описание программы	5
4 Схемы программы	7
4.1 Блок-схема программы.....	7
4.2 Блок-схема алгоритма.....	8
5 Тестирование программы.....	9
6 Отладка	10
7 Совместная разработка.....	11
Заключение	15
Список используемой литературы	16
Приложение А. Листинг программы.....	17

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. WEEEK — сервис для управления личными и командными проектами. В основе WEEEK 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 WEEEK 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 пунктов под буквами а-г выводится сообщение, в котором пользователю необходимо ввести количество значений для сортировки.

```
printf("Введите количество элементов массива: ");  
scanf("%d", &size);
```

После того, как данные были введены, генерируется массив из случайных чисел.

```
array = (int*)malloc(size * sizeof(int));  
for (int i = 0; i < size; i++)  
{  
    array[i] = rand() - rand();  
}
```

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.

Алгоритм для сортировки по возрастанию:

```
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;
        }
    }

```

Алгоритм для сортировки по убыванию:

```

    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;
        }
    }

```

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 разделе 4 на рисунках 1, 2.

Листинг программы приведен в приложении А.

4 Схемы программы

4.1 Блок-схема программы

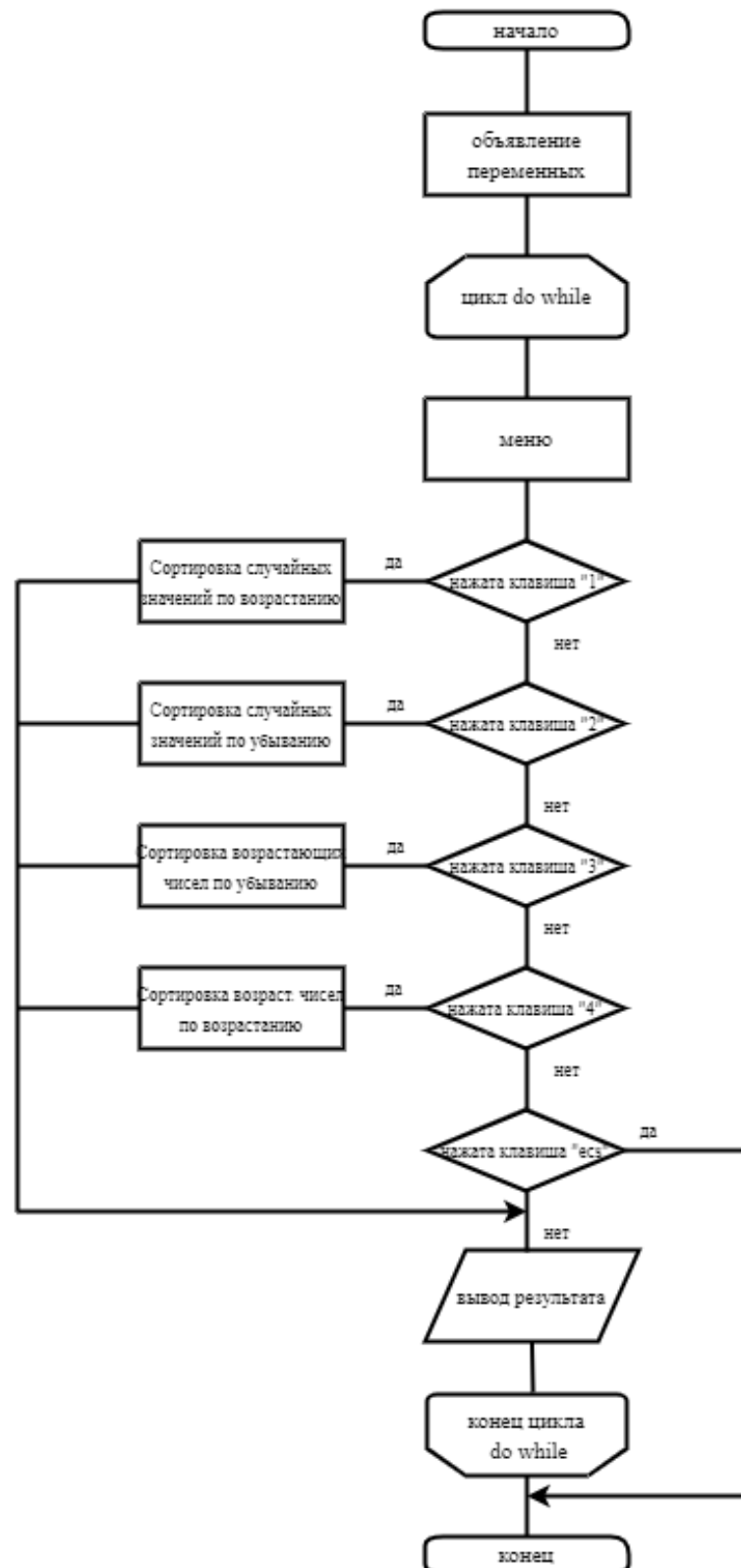


Рисунок 1 – Блок-схема программы

4.2 Блок-схема алгоритма

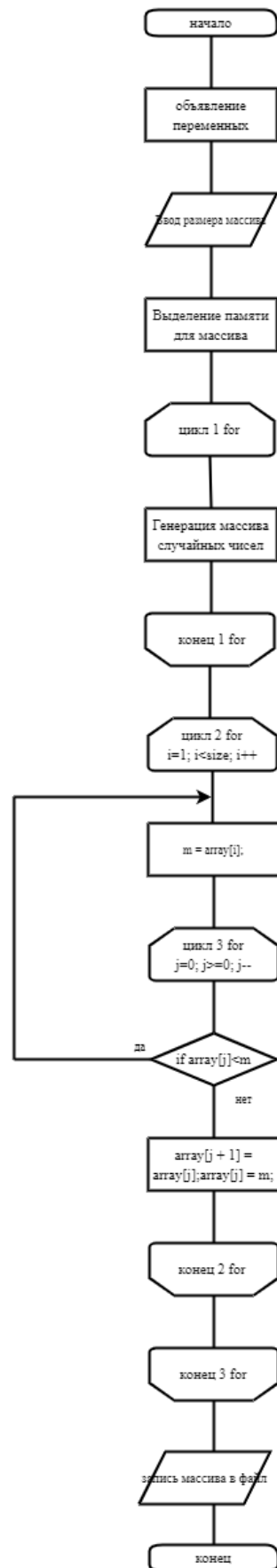


Рисунок 2 – Блок-схема алгоритма

5 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

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.

7 Совместная разработка

Для удобства совместной разработки был использован сервис WEEK.

Определили задачи проекта, назначили приоритет задачам.

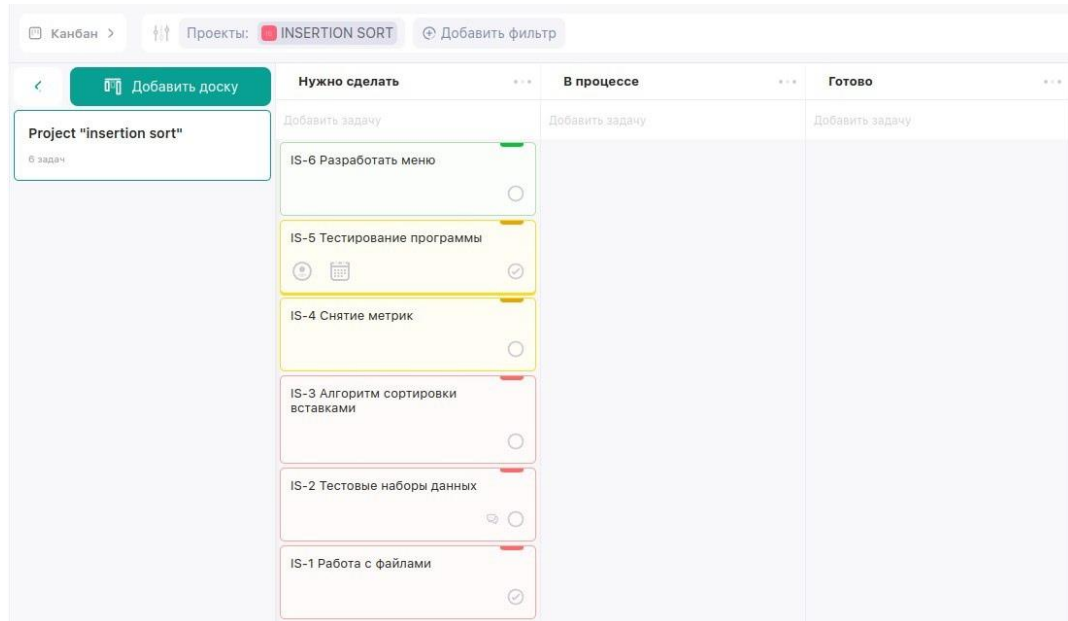


Рисунок 4 – Определение задач проекта

Разделили роли, назначили исполнителей задачам.

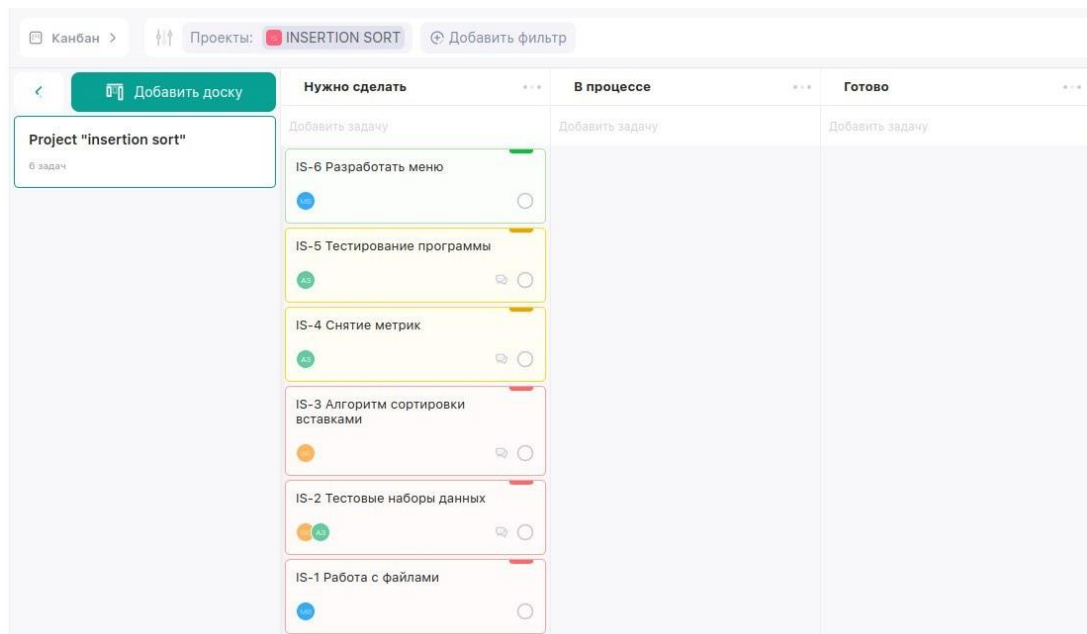


Рисунок 5 – Распределение задач проекта

Обсуждали выполнение задачи на канбан-доске.

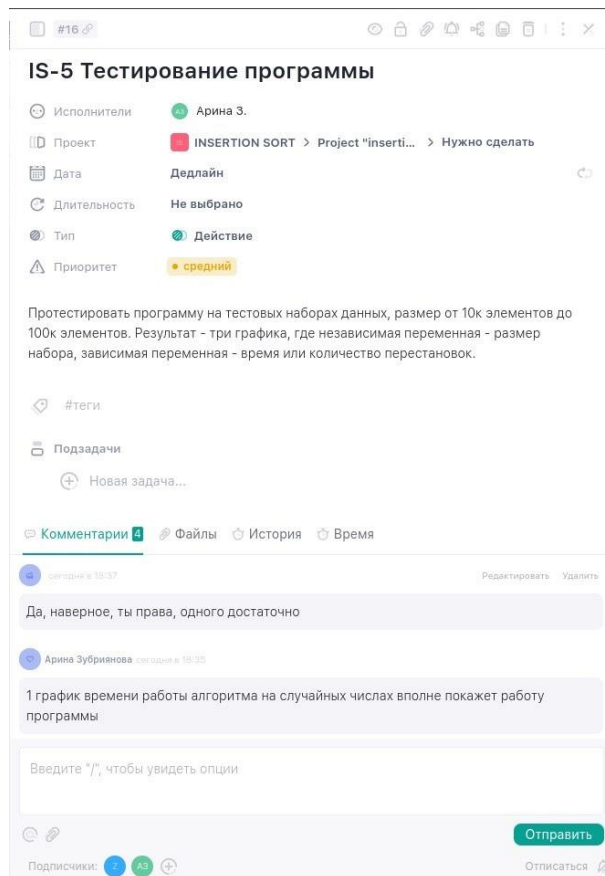


Рисунок 6 – Обсуждение задач проекта

Корректировали статус задач по мере выполнения.

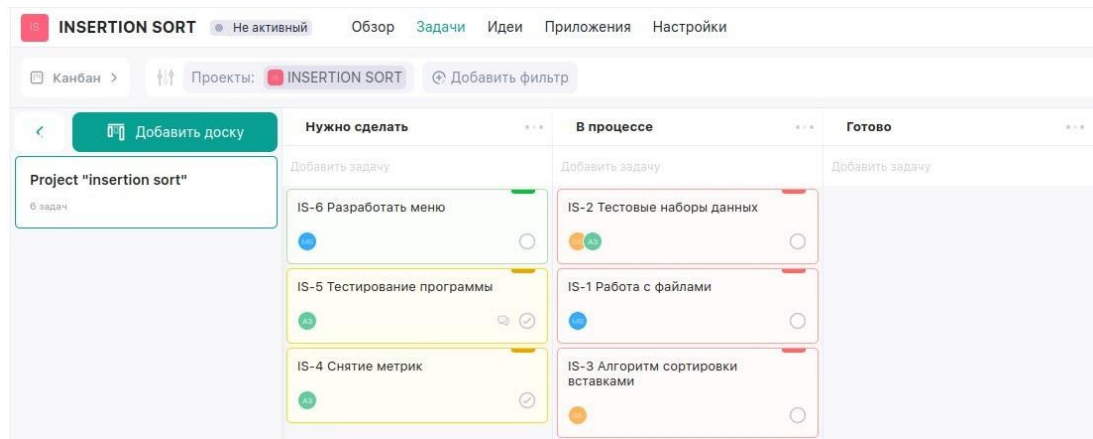
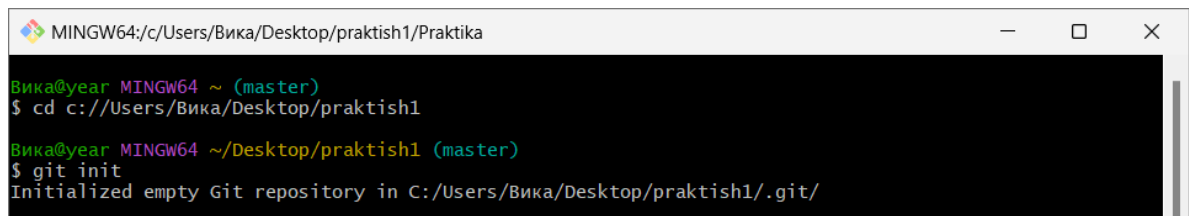


Рисунок 7 – Корректирование задач

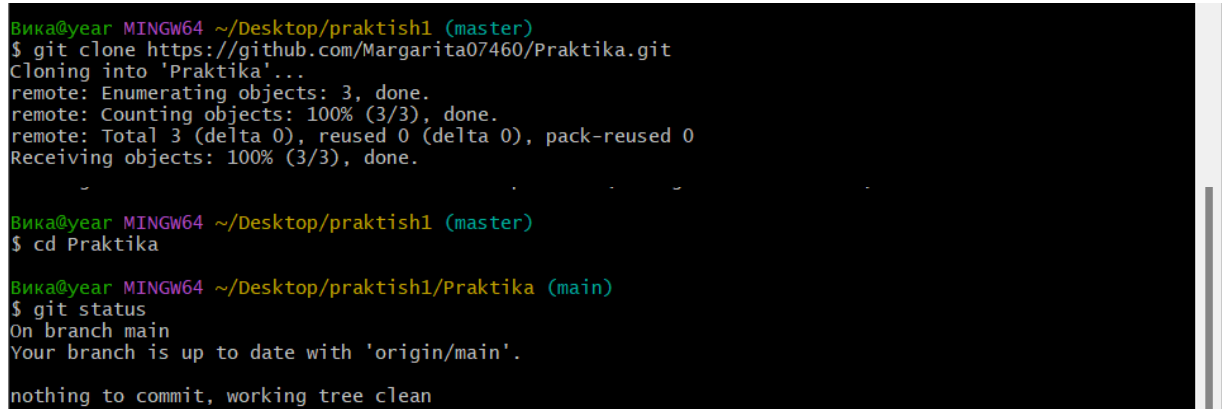
Во время работы над данной практикой наша бригада осуществляла совместную работу в GitHub.

Мною была написана программа сортировки, она была загружена на удаленный репозиторий Github, на ветку main.



```
MINGW64:/c:/Users/Вика/Desktop/praktish1/Praktika
Вика@year MINGW64 ~ (master)
$ cd c://Users/Вика/Desktop/praktish1
Вика@year MINGW64 ~/Desktop/praktish1 (master)
$ git init
Initialized empty Git repository in C:/Users/Вика/Desktop/praktish1/.git/
```

Рисунок 8 – Инициализация репозитория



```
Вика@year MINGW64 ~/Desktop/praktish1 (master)
$ git clone https://github.com/Margarita07460/Praktika.git
Cloning into 'Praktika'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

Вика@year MINGW64 ~/Desktop/praktish1 (master)
$ cd Praktika
Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 9 –Загрузка папки с удаленного репозитория

```

Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$ git add --all

Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$ git commit -m "алгоритм"
[main f1d53af] алгоритм
21 files changed, 352 insertions(+)
create mode 100644 Source.cpp
create mode 100644 praktish1.sln
create mode 100644 praktish1.vcxproj
create mode 100644 praktish1.vcxproj.filters
create mode 100644 praktish1.vcxproj.user
create mode 100644 x64/Debug/Source.obj
create mode 100644 x64/Debug/praktish1.exe
create mode 100644 x64/Debug/praktish1.exe.recipe
create mode 100644 x64/Debug/praktish1.ilkb
create mode 100644 x64/Debug/praktish1.log
create mode 100644 x64/Debug/praktish1.pdb
create mode 100644 x64/Debug/praktish1.tlog/CL.command.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/CL.read.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/CL.write.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/link.command.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/link.read.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/link.write.1.tlog
create mode 100644 x64/Debug/praktish1.tlog/praktish1.lastbuildstate
create mode 100644 x64/Debug/praktish1.vcxproj.FileListAbsolute.txt
create mode 100644 x64/Debug/vc143.idb
create mode 100644 x64/Debug/vc143.pdb

Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$ git log
commit f1d53af8db8b11b60e7d7d73d0b1b57e8902e8ca (HEAD -> main)
Author: Vika <vikka_kondrateva@mail.ru>
Date: Sat Jul 1 20:11:56 2023 +0300

    алгоритм

commit ef17bbc961b044e2158b61a5aec19ce395dbf7ab (origin/main, origin/HEAD)
Author: Margarita07460 <132387176+Margarita07460@users.noreply.github.com>
Date: Sat Jul 1 19:18:57 2023 +0300

    Initial commit

Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$ git push
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 12 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (26/26), 298.72 KiB | 3.21 MiB/s, done.
Total 26 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/Margarita07460/Praktika.git
ef17bbc..f1d53af main -> main

Вика@year MINGW64 ~/Desktop/praktish1/Praktika (main)
$

```

Рисунок 10 – Загрузка программы на удалённый репозиторий

Ссылка на удаленный репозиторий:

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

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, осуществляющий сортировку вставками для массива случайно сгенерированных и уже отсортированных чисел, оформлен отчет по данной практике.

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)


```

        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

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

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;

```

сортировки

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

    }

    fclose(f);

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

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

    time_t start = clock();                //время до

    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

    for (int i = 0; i < size; i++) {
        fprintf(f, "%d  ", array[i]);

    }
```

сортировки

```

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);

```

сортировки

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

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

time_t start = clock();                                //время до

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

for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", array[i]);
}
fclose(f);
double time = (stop - start) / 1000.0;                //время

printf("\n");
printf("Время выполнения сортировки: ");
printf("%lf\n", time);
printf("Количество перестановок: ");
```

сортировки

```
        printf("%o\n", count);

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