

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)

_____ Картушин Роман Алексеевич _____

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)

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)

Картушин Роман Алексеевич

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)

Картушин Роман Алексеевич

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»

Картушин Р.А. выполнял практическое задание «Сортировка Шелла». На первоначальном этапе были изучен и проанализирован алгоритм сортировки Шелла, был выбран метод решения и язык программирования С, на котором была написана программа сортировки массива. Также, осуществил работу с файлами. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Картушин Р.А. " " 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)

Картушин Роман Алексеевич

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»

В процессе выполнения практики Картушин Р.А. решал следующие задачи: создание алгоритма сортировки Шелла, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии сортировки Шелла, реализован метод работы с файлами. Во время выполнения работы Картушин Р.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 и сортировке.

За выполнение работы Картушин Р.А. заслуживает оценки « ».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2025 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	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 образуют отдельный класс алгоритмов, применяются практически во всех задачах обработки информации. При этом они настолько тесно связаны друг с другом, что образуют отдельный класс алгоритмов. Алгоритмы сортировки, как правило, применяются с целью осуществления последующего более быстрого поиска. Например, трудно пользоваться словарями, если бы слова в них не были бы упорядочены по алфавиту.

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 Шелла представляет собой улучшенную версию сортировки вставками, отличающуюся значительно более высокой производительностью благодаря использованию переменного шага сравнения элементов. Этот алгоритм, хотя и является нестабильным, демонстрирует на практике существенно лучшие результаты по сравнению с простыми методами вроде сортировки выбором или пузырьковой сортировки.

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 Шелла 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 Шелла

- хорошо работает на массивах среднего размера;
- не требует дополнительной памяти;
- работает быстрее простых алгоритмов.

1.2 Недостатки алгоритма сортировки Шелла

- нестабильная сортировка;
- ограниченная применимость;
- сложность оптимизации.

1.3 Типичные сценарии применения данного алгоритма

- сортировка средних по размеру массивов (когда быстрая сортировка или сортировка слиянием избыточны);
- предварительная сортировка в гибридных алгоритмах (например, перед финальной сортировкой вставками);
- обработка частично упорядоченных данных (например, логов, временных рядов, почти отсортированных массивов).

2 Выбор решения

Нашей бригадой было выбрано вести разработку в среде Microsoft Visual Studio на языке C.

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.

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 GitHub и GitHub Projects. GitHub Projects — это встроенный в GitHub сервис для управления личными и командными проектами. Он поддерживает канбан-доски, дорожные карты, таблицы и т. д. GitHub Projects отлично подходит для организации совместной работы, помогая структурировать задачи, распределять нагрузку и контролировать прогресс.

3 Описание программы

При запуске программы выводится меню из пяти пунктов (рис. 1):

- а) сортировка случайных значений по возрастанию;
- б) сортировка случайных значений по убыванию;
- в) сортировка возрастающих значений по убыванию;
- г) Ввод своего массива для сортировки;
- д) esc – выход.

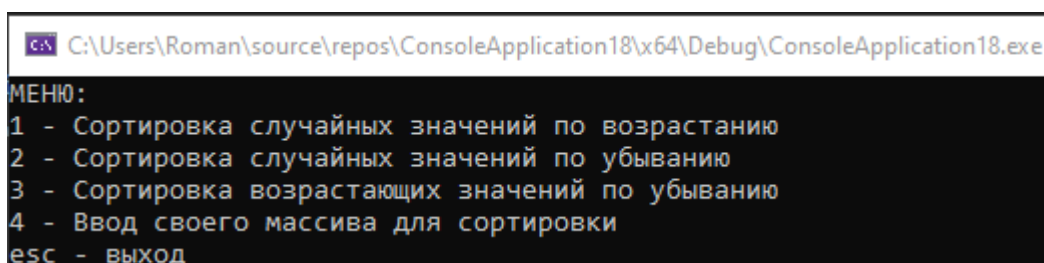


Рисунок 1 – Меню.

Пользователь должен выбрать тот пункт, который ему требуется. При выборе вариантов а-в выводится сообщение, в котором пользователю необходимо ввести количество значений для сортировки (рис. 2).

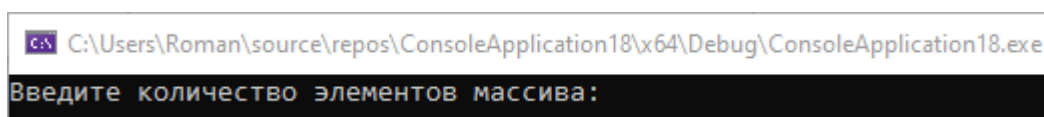


Рисунок 2 – Приглашение для ввода желаемого количества сортируемых значений

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

Программа выполняет сортировку Шелла, которая последовательно упорядочивает массив, сравнивая и перемещая элементы на уменьшающихся расстояниях. Алгоритм проходит по массиву несколько раз, на каждом этапе улучшая порядок элементов, пока не будет достигнута полная сортировка.

При выборе варианта “г”, пользователю предлагается ввести количество элементов исходного массива и содержимое массива (рис. 3).

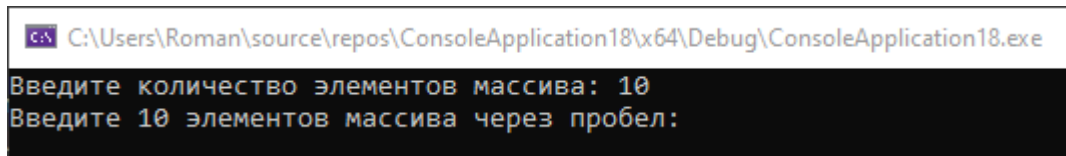


Рисунок 3 – Составление массива для сортировки.

После завершения сортировки программа выводит время работы, количество выполненных перестановок, а также исходный и отсортированный массив, что позволяет оценить эффективность алгоритма для разных типов данных (рис. 4).

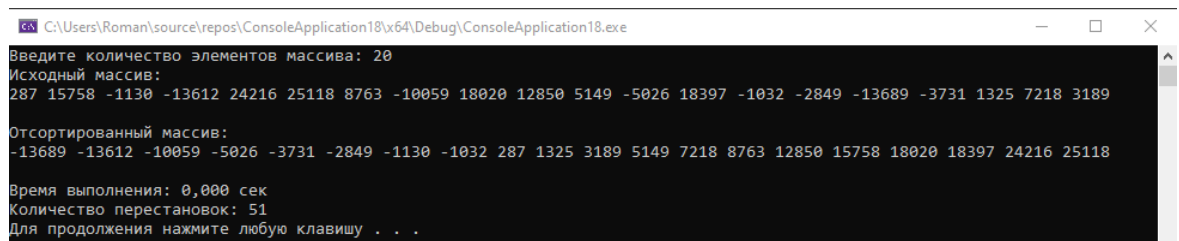


Рисунок 4 -Результат сортировки.

При выборе варианта “д”, программа завершается.

В программе реализованы 3 функции:

1. PrintArray – функция, отвечающая за вывод на экран исходного и отсортированного массивов.
2. shellSort – функция, в которой реализован алгоритм сортировки
3. main – главная функция, в которой реализован цикл do-while, с подсчетом времени сортировки, количества перестановок и т.д

Листинг программы представлен в приложении А.

4. Схемы программы

4.1 Блок-схема программы

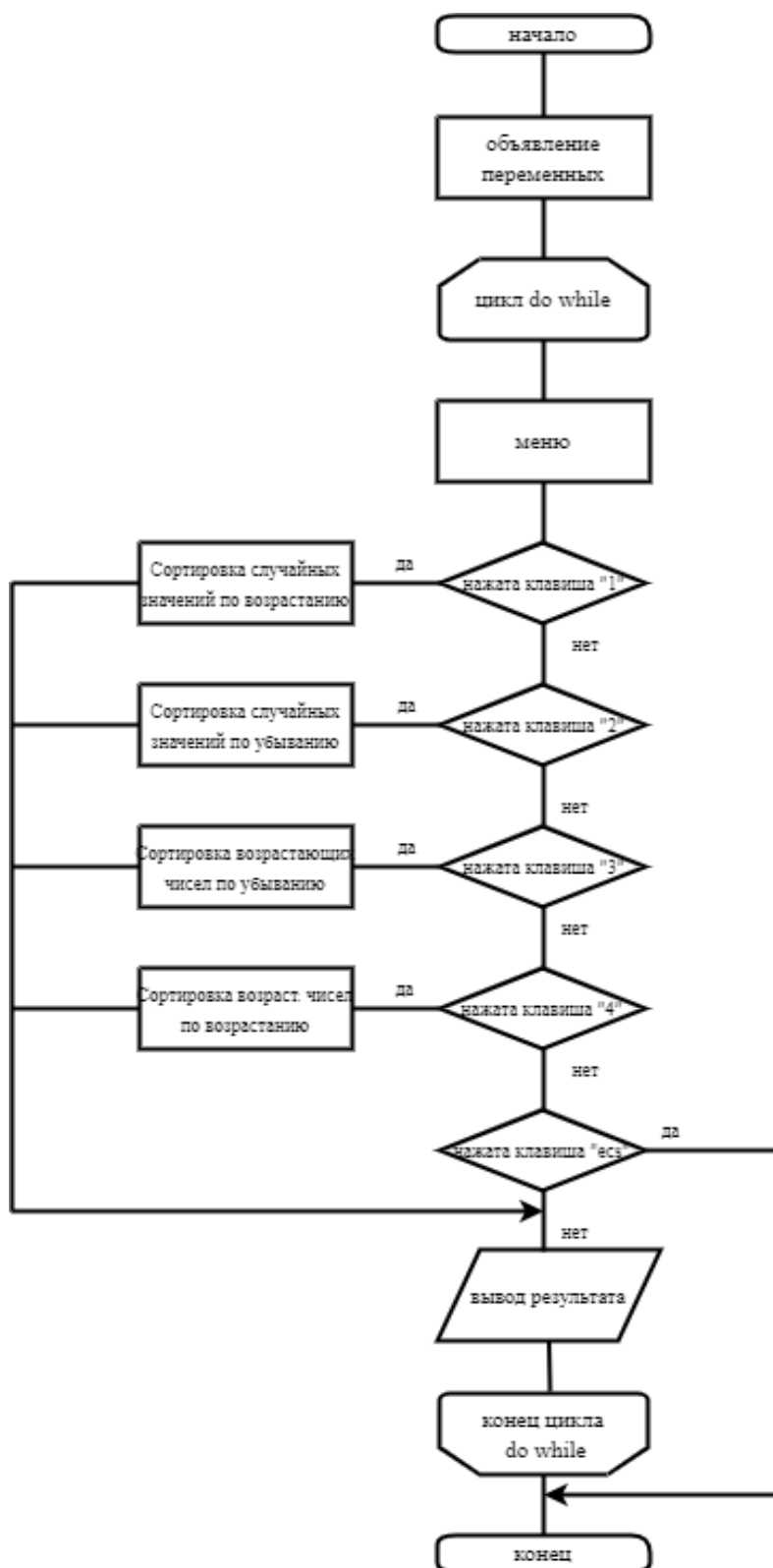


Рисунок 5 - Блок-схема программы

4.2 Блок-схема алгоритма

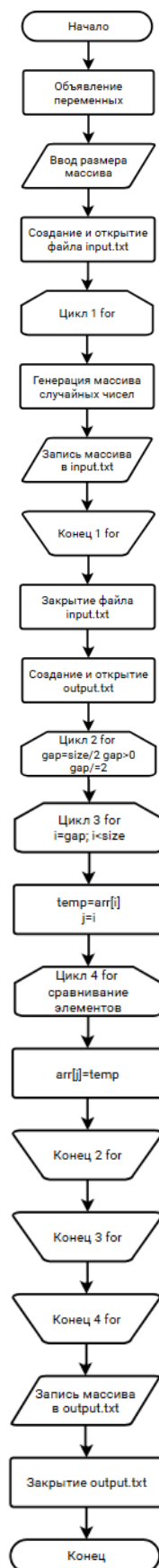


Рисунок 6 - Блок-схема алгоритма

5 Тестирование программы

5.1 Тестирование на разных наборах данных

Тестирование программы на разных наборах исходных данных представлено в Таблице 1.

Таблица 1 – Тестовый набор данных

№ Теста	Размер массива	Время выполнения сортировки в секундах	Количество перестановок
1	10000	0.001	217153
2	50000	0.010	1591378
3	100000	0.023	3622385
4	300000	0.077	12723715
5	500000	0.135	24144151
6	1000000	0.299	57220199
7	3000000	0.997	190487746

5.2 Анализ полученных результатов тестирования (анализ работы алгоритма)

На основании анализа данных, полученных в результате тестирования алгоритма сортировки Шелла, 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.

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.

Отладка проходила в рабочем процессе, в процессе разработки и после завершения написания программы.

7 Совместная разработка

Для удобства совместной разработки был использован сервис GitHub Projects. Определили задачи проекта, назначили приоритет задачам (рис.4).

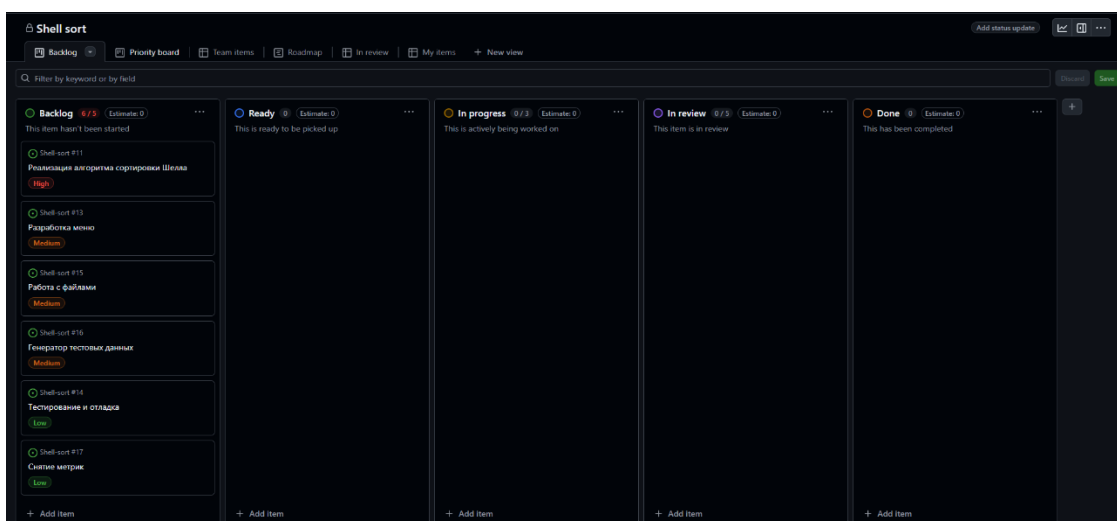


Рисунок 4 – Определение задач проекта.

Разделили роли, назначили исполнителей задач (рис. 5).

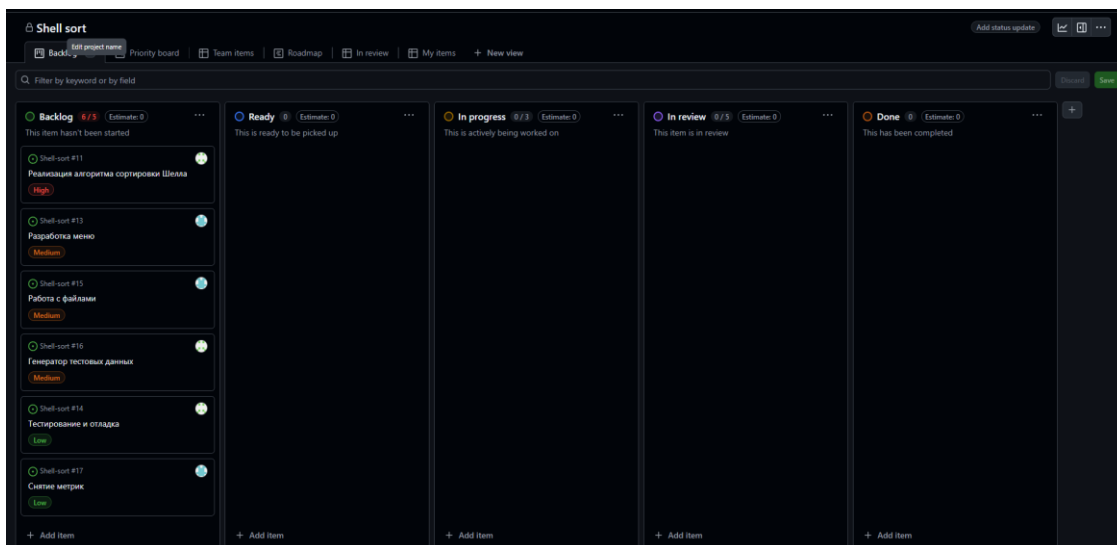


Рисунок 5 – Распределение задач проекта.

Корректировали статус задач по мере выполнения (рис.6).

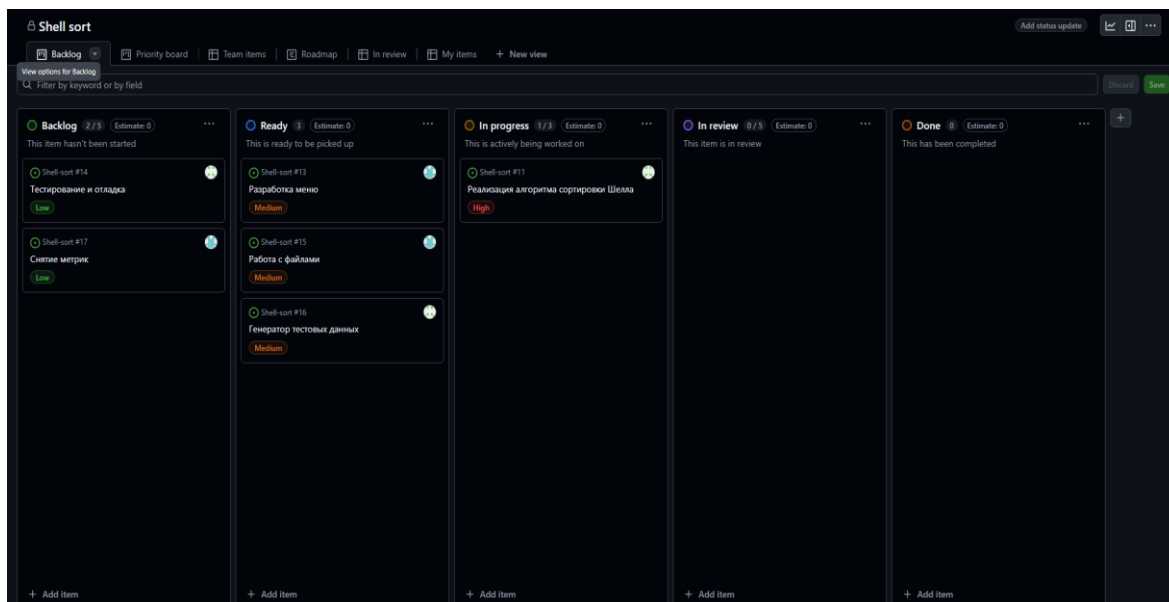


Рисунок 6 – Корректировка задач.

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

Мною была написан алгоритм сортировки и реализовано меню для выбора варианта сортировки, проект была загружен на удаленный репозиторий Github, на ветку main.

На рисунке 7 представлены коммиты созданные бригадой.

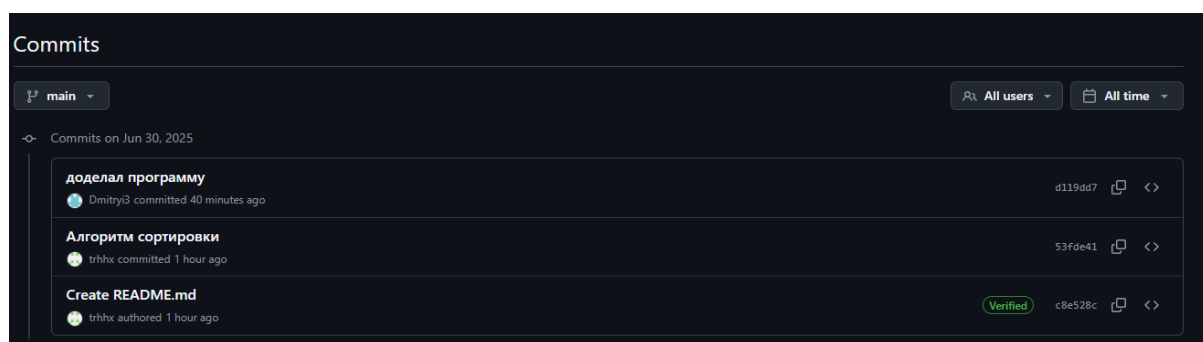


Рисунок 7 – Созданные коммиты.

На рисунке 8 представлена ветка main после всех доработок программы.

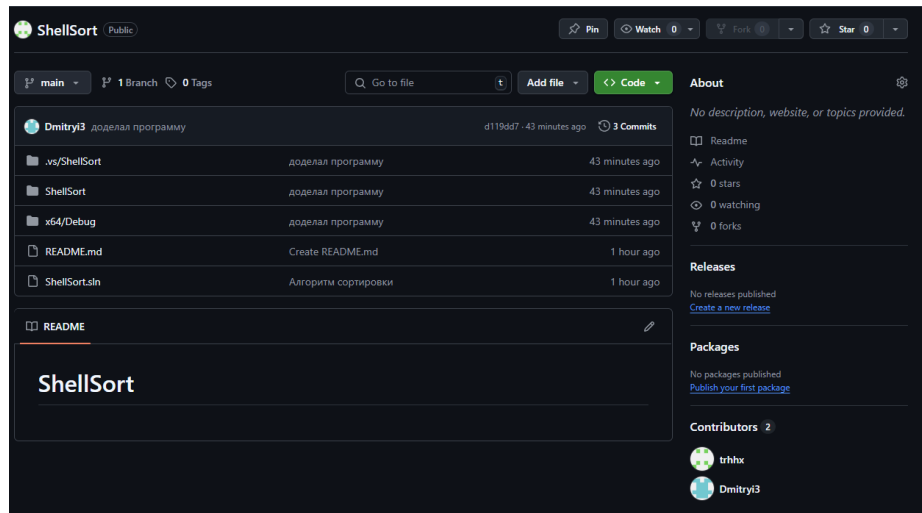


Рисунок 8 – Ветка main.

Для загрузки данных на локальный репозиторий, а также для их отправки на удаленный репозиторий использовался Git Bash.

На рисунке 9 представлено использование команды `git clone` для клонирования удалённого репозитория в локальный.

```

MINGW64:/c/Users/Roman/Desktop

Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop (master)
$ git init
Reinitialized existing Git repository in C:/Users/Roman/Desktop/.git/

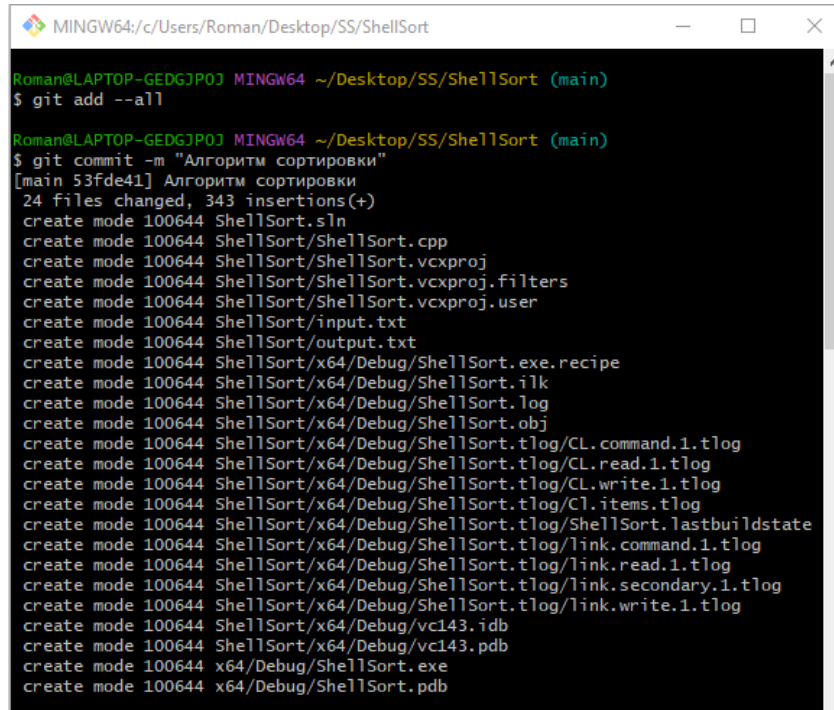
Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop (master)
$ git clone https://github.com/trhxx/ShellSort.git
Cloning into 'ShellSort'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop (master)
$

```

Рисунок 9 – Клонирование репозитория.

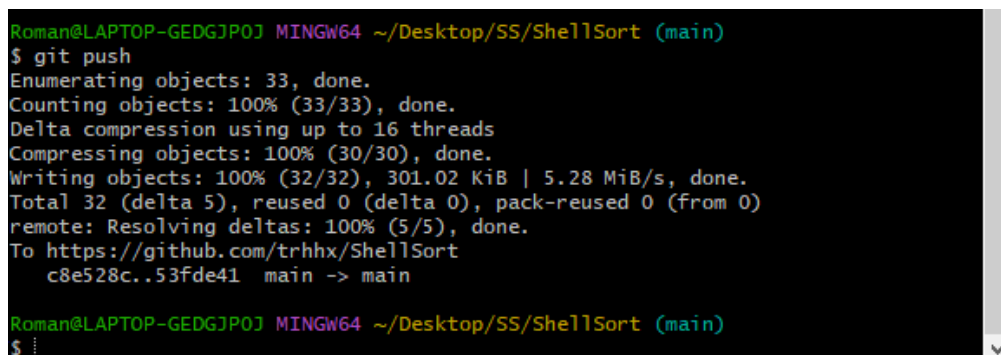
На рисунке 10 показано добавление файлов программы в репозиторий и создание соответствующего коммита.



```
MINGW64: c:/Users/Roman/Desktop/SS/ShellSort
Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop/SS/ShellSort (main)
$ git add --all
Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop/SS/ShellSort (main)
$ git commit -m "Алгоритм сортировки"
[main 53fde41] Алгоритм сортировки
24 files changed, 343 insertions(+)
create mode 100644 ShellSort.sln
create mode 100644 ShellSort/ShellSort.cpp
create mode 100644 ShellSort/ShellSort.vcxproj
create mode 100644 ShellSort/ShellSort.vcxproj.filters
create mode 100644 ShellSort/ShellSort.vcxproj.user
create mode 100644 ShellSort/input.txt
create mode 100644 ShellSort/output.txt
create mode 100644 ShellSort/x64/Debug/ShellSort.exe.recipe
create mode 100644 ShellSort/x64/Debug/ShellSort.ilkg
create mode 100644 ShellSort/x64/Debug/ShellSort.log
create mode 100644 ShellSort/x64/Debug/ShellSort.obj
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/CL.command.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/CL.read.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/CL.write.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/CL.items.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/ShellSort.lastbuildstate
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/link.command.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/link.read.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/link.secondary.1.tlog
create mode 100644 ShellSort/x64/Debug/ShellSort.tlog/link.write.1.tlog
create mode 100644 ShellSort/x64/Debug/vc143.idb
create mode 100644 ShellSort/x64/Debug/vc143.pdb
create mode 100644 x64/Debug/ShellSort.exe
create mode 100644 x64/Debug/ShellSort.pdb
```

Рисунок 10 – Создание коммита.

На рисунке 11 представлено использование команды git push для отправки данных на удаленный репозиторий.



```
Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop/SS/ShellSort (main)
$ git push
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 16 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (32/32), 301.02 KiB | 5.28 MiB/s, done.
Total 32 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/trhhx/ShellSort
   c8e528c..53fde41  main -> main
Roman@LAPTOP-GEDGJP0J MINGW64 ~/Desktop/SS/ShellSort (main)
$
```

Рисунок 11 – Отправка данных на удаленный репозиторий.

Ссылка на удалённый репозиторий - <https://github.com/trhhx/ShellSort>

Заключение

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

Мною был написан алгоритм сортировки программы и создано меню программы, позволяющее выбрать тип сортировки.

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 Шелла [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Сортировка_Шелла

Приложение А. Листинг программы

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

// Функция вывода массива
void printArray(int* arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// Функция сортировки Шелла
void shellSort(int* arr, int size, int srt, long* count) {
    *count = 0; // Обнуляем счетчик перестановок

    // Основной цикл с уменьшающимся шагом
    for (int gap = size / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < size; i++) {
            int temp = arr[i];
            int j;

            // Сортировка по возрастанию
            if (srt) {
                for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
                    arr[j] = arr[j - gap];
                    (*count)++;
                }
            }

            // Сортировка по убыванию
            else {

```

```

        for (j = i; j >= gap && arr[j - gap] < temp; j -= gap) {
            arr[j] = arr[j - gap];
            (*count)++;
        }
    }
    arr[j] = temp;
    if (j != i) (*count)++;
}
}
}

```

```

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

    FILE* f;
    int size;
    int* arr;
    char ch;
    long count = 0;

    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 - Ввод своего массива для сортировки\n");
        printf("esc - выход\n");
        ch = _getch();

        switch (ch) {
            case '1': {
                // Случайные числа по возрастанию
                system("cls");
                printf("Введите размер массива: ");
            }

```



```

scanf("%d", &size);
arr = (int*)malloc(size * sizeof(int));

// Генерация и запись случайных чисел
f = fopen("input.txt", "w");
printf("Исходный массив:\n");
for (int i = 0; i < size; i++) {
    arr[i] = rand() - rand();
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

// Сортировка и замер времени
time_t start = clock();
shellSort(arr, size, 1, &count);
time_t stop = clock();

// Запись результата
f = fopen("output.txt", "w");
printf("\nОтсортированный массив:\n");
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

printf("\nВремя: %.3lf сек\n", (stop - start) / 1000.0);
printf("Перестановок: %ld\n", count);

free(arr);
system("pause");
break;
}

case '2': {

```

```

// Случайные числа по убыванию (аналогично case 1, но srt=0)
system("cls");
printf("Введите размер массива: ");
scanf("%d", &size);
arr = (int*)malloc(size * sizeof(int));

f = fopen("input.txt", "w");
printf("Исходный массив:\n");
for (int i = 0; i < size; i++) {
    arr[i] = rand() - rand();
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

time_t start = clock();
shellSort(arr, size, 0, &count);
time_t stop = clock();

f = fopen("output.txt", "w");
printf("\nОтсортированный массив:\n");
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

printf("\nВремя: %.3lf сек\n", (stop - start) / 1000.0);
printf("Перестановок: %ld\n", count);

free(arr);
system("pause");
break;
}

case '3': {

```

```

// Упорядоченный массив по убыванию
system("cls");

printf("Введите размер массива: ");
scanf("%d", &size);
arr = (int*)malloc(size * sizeof(int));

// Генерация возрастающего массива
f = fopen("input.txt", "w");
printf("Исходный массив:\n");
arr[0] = rand() % 1000;
for (int i = 1; i < size; i++) {
    arr[i] = arr[i - 1] + rand() % 100 + 1;
}
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

// Сортировка по убыванию
time_t start = clock();
shellSort(arr, size, 0, &count);
time_t stop = clock();

f = fopen("output.txt", "w");
printf("\nОтсортированный массив:\n");
for (int i = 0; i < size; i++) {
    fprintf(f, "%d ", arr[i]);
}
printArray(arr, size);
fclose(f);

printf("\nВремя: %.3lf сек\n", (stop - start) / 1000.0);
printf("Перестановок: %ld\n", count);

free(arr);

```

```

        system("pause");
        break;
    }

case '4': {
    // Ручной ввод массива
    system("cls");
    printf("Введите размер массива: ");
    scanf("%d", &size);
    arr = (int*)malloc(size * sizeof(int));

    printf("Введите %d элементов:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Выбор направления сортировки
    printf("Выберите направление (1 - возр., 2 - убыв.): ");
    int srt1 = _getch() == '1' ? 1 : 0;

    f = fopen("input.txt", "w");
    printf("\nИсходный массив:\n");
    for (int i = 0; i < size; i++) {
        fprintf(f, "%d ", arr[i]);
    }
    printArray(arr, size);
    fclose(f);

    time_t start = clock();
    shellSort(arr, size, srt1, &count);
    time_t stop = clock();

    f = fopen("output.txt", "w");
    printf("\nОтсортированный массив:\n");
    for (int i = 0; i < size; i++) {
        fprintf(f, "%d ", arr[i]);
    }
}

```

```

    }
    printArray(arr, size);
    fclose(f);

    printf("\nВремя: %.3lf сек\n", (stop - start) / 1000.0);
    printf("Перестановок: %ld\n", count);

    free(arr);
    system("pause");
    break;
}
}
} while (ch != 27);
return 0;
}

```