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)

Юсупов Владислав Ренатович

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)

Юсупов Владислав Ренатович

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 (распределенная практика)	26	20.06.25 – 24.06.25	
2	Подбор и изучение материала по теме работы (распределенная практика)	26	24.06.25 – 26.06.25	
3	Разработка алгоритма (распределенная практика)	26	26.06.25 – 28.06.25	
4	Описание алгоритма и программы (сосредоточенная практика)	30	04.07.25 – 09.07.25	
5	Тестирование (сосредоточенная практика)	30	09.07.25 – 12.07.25	
6	Получение и анализ результатов (сосредоточенная практика)	38	12.07.25 – 14.07.25	

7	Оформление отчёта (сосредоточенная практика)	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)

Юсупов Владислав Ренатович

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 – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Юсупов В.Р. 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 методом Шелла. Также, осуществил работу с методом заполнения массива случайными значениями для сортировки. Оформил отчёт.

Специалист Юсупов В.Р. "___" _____ 2025 г.

Руководитель практики Карамышева Н.С. "___" _____ 2025 г.

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)

Юсупов Владислав Ренатович

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 Юсупов Владислав Ренатович решал следующие задачи: разработка алгоритма сортировки Шелла, 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 сортировки Шелла, получены результаты работы алгоритма, сделаны выводы эффективности работы алгоритма сортировки Шелла. Во время выполнения работы Юсупов Владислав Ренатович 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 Юсупов Владислав Ренатович заслуживает оценки «_____».

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

Содержание

Введение

Сортировка данных на сегодняшний день является одним из наиболее распространенных процессов современной обработки данных. В профессиональной деятельности часто можно встретить задачу, которая подразумевает сортировку данных различными способами и методами.

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 приемы и методы построения алгоритмов. 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.

Преимущество сортировки Шелла заключается в том, что она позволяет ускорить процесс упорядочивания данных за счёт уменьшения расстояния между сравниваемыми элементами на каждом этапе алгоритма. Это делает её более предпочтительной для использования в реальных приложениях, где требуется сортировка больших массивов 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 Шелла 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 Шелла

- не используется дополнительная память;
- работает быстрее простых алгоритмов сортировки;
- хорош в использовании для средних объемов данных. На небольших и средних массивах может быть даже быстрее, чем *QuickSort.*

1.2 Недостатки алгоритма сортировки Шелла

- сложнее в реализации, чем другие алгоритмы;
- не всегда предсказуем. Скорость работы сильно зависит от входных данных;
- сложность зависит от выбора шага. В худшем случае может деградировать до $O(n^2)$, если шаги выбраны неудачно.

1.3 Типичные сценарии применения данного алгоритма

- обучение основам алгоритмизации и программирования в образовательных целях;

- сортировка контактов или сообщений в мессенджерах, где важна скорость реакции на пользовательский ввод;

- сортировка небольших объемов данных.

2 Выбор решения

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. Этот язык был выбран сразу, потому что обучение программированию проходило именно на языке Си.

В качестве среды программирования была выбрана программа *Microsoft Visual Studio Code*. Это удобный и современный редактор кода для разных языков программирования.

Для удобства совместной разработки было использовано приложение Discord. Это бесплатное приложение, которое поддерживает опции голосового, видео- и текстового чатов. Оно доступно в десктопной, мобильной и веб-версии в браузере. С помощью приложения все участники бригады общались и помогали друг другу.

3 Описание программы

При запуске программы выводится меню из трех пунктов:

1) ввод значений массива вручную;
2) чтение элементов массива из файла *input.txt*;
3) создание файла с массивом случайных чисел для чтения;

Пользователю требуется выбрать тот пункт, который ему требуется. При выборе первого варианта выводится сообщение, в котором пользователю необходимо ввести содержимое массива через пробел.После того, как данные были введены, созданный массив сортируется методом Шелла и итоговый его вид записывается в файл *output.txt*. Параллельно происходит подсчёт времени на сортировку. Сама сортировка происходит посредством функции *shellSort*.

При выборе второго варианта происходит заполнение массива из файла. Затем программа занимается сортировкой Шелла посредством функции *shellSort* и подсчитывает время её выполнения. В случае ошибки программа выводит соответствующее сообщение.

При выборе третьего варианта пользователя просят ввести размер случайного массива. Затем происходит функция *createFile*. Затем начинается выполнение функции readFile и если ошибки не произошло, то начинается выполнение сортировки Шелла посредством функции shellSort с подсчётом времени на её выполнение. Сгенерированный случайный массив записывается в файл *input.txt*, отсортированная версия в *output.txt*.

3.1 Описание личного вклада в проект

Моя задача заключалась в разработке и интеграции алгоритма метода заполнения массива случайными значениями для последующей сортировки. Для этого мною была создана часть оператора switch под номером „3“.

Алгоритм случайного заполнения заключается в вводе размера массива и использования подфункций readFile и CreateFile, после чего полученный массив передаётся другим разработчикам для проведения сортировки.

```
switch (operation)
{
 case '3': {
    int sizeArr;
    printf("Введите размер случайного массива:\n");
    scanf("%d", &sizeArr);
    createFile(sizeArr);

    int* arrayNum = readFile(&sizeArr);
    if (arrayNum != NULL) {
        time_t startTime = clock(); // Начало отсчёта
        shellSort(arrayNum, sizeArr); // Сортировка массива
        time_t endTime = clock(); // Конец отсчёта

        outputFile(arrayNum, sizeArr);
        printf("Сгенерированный случайный массив записан в файл input.txt,\n а отсортированный его вариант записан в файл output.txt");
```

3.1 Описание личного вклада в проект

Моя задача заключалась в разработке и интеграции алгоритма пользовательского интерфейса, а так же метода заполнения файла случайными числами для последующей сортировки.

Алгоритм пользовательского интерфейса заключается в выводе меню выбора действий, и в соответствии с выбором выполнения определённых действий, ввод массива вручную и его сортировки, считывании массива из файла с его сортировкой, генерацию массива случайных чисел.

```
printf("Выберите действие:\n");
printf("1: Ввод массива вручную.\n");
printf("2: Считать массив из файла ввода input.txt.\n");
printf("3: Создать файл с массивом случайных чисел.\n");
char operation = getchar();
getchar();
switch (operation) {
case '1': {
    printf("Введите массив чисел через пробел:\n");
    int currentCount = 0, sizeArray = 10;
    int *arrayNum = malloc(sizeArray * sizeof(int));
    char c;

    while (scanf("%d%c", &arrayNum[currentCount++], &c) == 2) {
        if (currentCount >= sizeArray) {
            sizeArray *= 2;
            arrayNum = realloc(arrayNum, sizeArray * sizeof(int));
        }
        if (c == '\n') break;
    }
    arrayNum = realloc(arrayNum, --currentCount * sizeof(int));

    time_t startTime = clock();; // начальное время
    shellSort(arrayNum, currentCount); // сортировка массива
    time_t endTime = clock();; // конечное время

    outputFile(arrayNum, currentCount);
    printf("Отсортированный массив сохранён в файл output.txt");

    double totalTime = (double)(endTime - startTime) / CLOCKS_PER_SEC; // общее в
    printf("\nВремя сортировки: %.3f\n", totalTime);
```

```
        free(arrayNum);
        break;
    }
    case '2': {
        int sizeArr = NULL;
        int* arrayNum = readFile(&sizeArr);
        if (arrayNum != NULL) {
            time_t startTime = clock();; // начальное время
            shellSort(arrayNum, sizeArr); // сортировка массива
            time_t endTime = clock();; // конечное время

            outputFile(arrayNum, sizeArr);
            printf("Отсортированный массив сохранён в файл output.txt");

            double totalTime = (double)(endTime - startTime) / CLOCKS_PER_SEC; //
общее время сортировки
            printf("\nВремя сортировки: %.3f\n", totalTime);}
        else printf("Ошибка чтения файла.");
        break; }
    case '3': {
        int sizeArr;
        printf("Введите размер желаемого массива:\n");
        scanf("%d", &sizeArr);
        createFile(sizeArr);

        int* arrayNum = readFile(&sizeArr);
        if (arrayNum != NULL) {
            time_t startTime = clock();; // начальное время
            shellSort(arrayNum, sizeArr); // сортировка массива
            time_t endTime = clock();; // конечное время

            outputFile(arrayNum, sizeArr);
            printf("Сгенерированный массив случайных чисел сохранён в файл
input.txt,\n а отсортированный вид данного массива сохранён в файл
output.txt");

            double totalTime = (double)(endTime - startTime) / CLOCKS_PER_SEC; //
общее время сортировки
            printf("\nВремя сортировки: %.3f\n", totalTime);
        }
        else printf("Ошибка чтения файла.");
        break;}
    default:
        break;}
```

Работа программы:

1. Программа выводит меню выбора действий для пользователя.
2. В зависимости от выбора пользователя выполняется определённая последовательность действий сопровождающаяся сортировкой массива,Перед и после сортировки проверяется текущее время из которых высчитывается итоговое время сортировки.
3. Пользователю выводится время сортировки массива, освобождает память выделенную под массив и файл с отсортированным массивом.

```
int createFile(int sizeArray) {

   FILE* file = fopen("input.txt", "w");

   if (file == NULL) {

      printf("\nERROR: file creation error\n");
 1.
      fclose(file);

      return 1;

   }


   srand(time(NULL));


   for (int i = 0; i < sizeArray; i++) fprintf(file, "%d\n", rand());


   fclose(file);


   return 0;

}
```

Работа функции:

1. Сначала программа открывает файл input.txt для редактирования или создаёт новый, если его нет в наличии.

2. Затем определяет размер массива чисел и заполняет его случайными числами, записывая их в файл.
3. И завершает свою работу, передавая 0 основной программе и закрывая файл.

4 Схемы программы

4.1 Блок-схема алгоритма

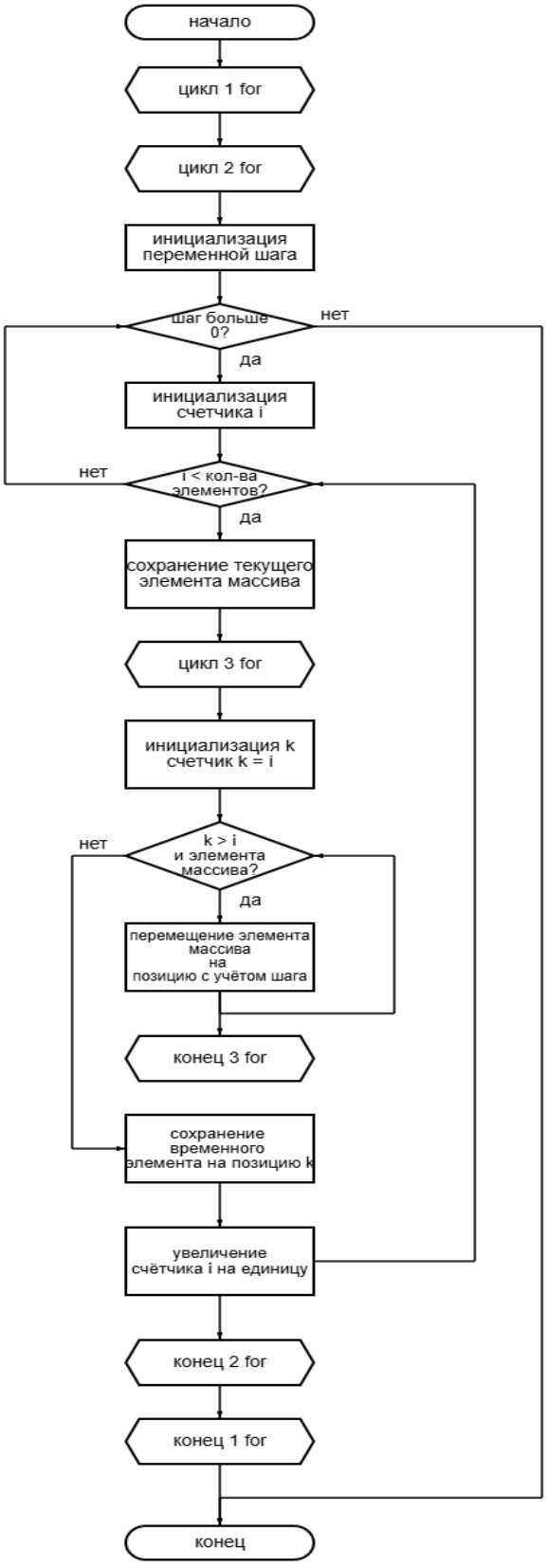

Рисунок 1 — Блок-схема алгоритма

4.2 Блок-схема программы

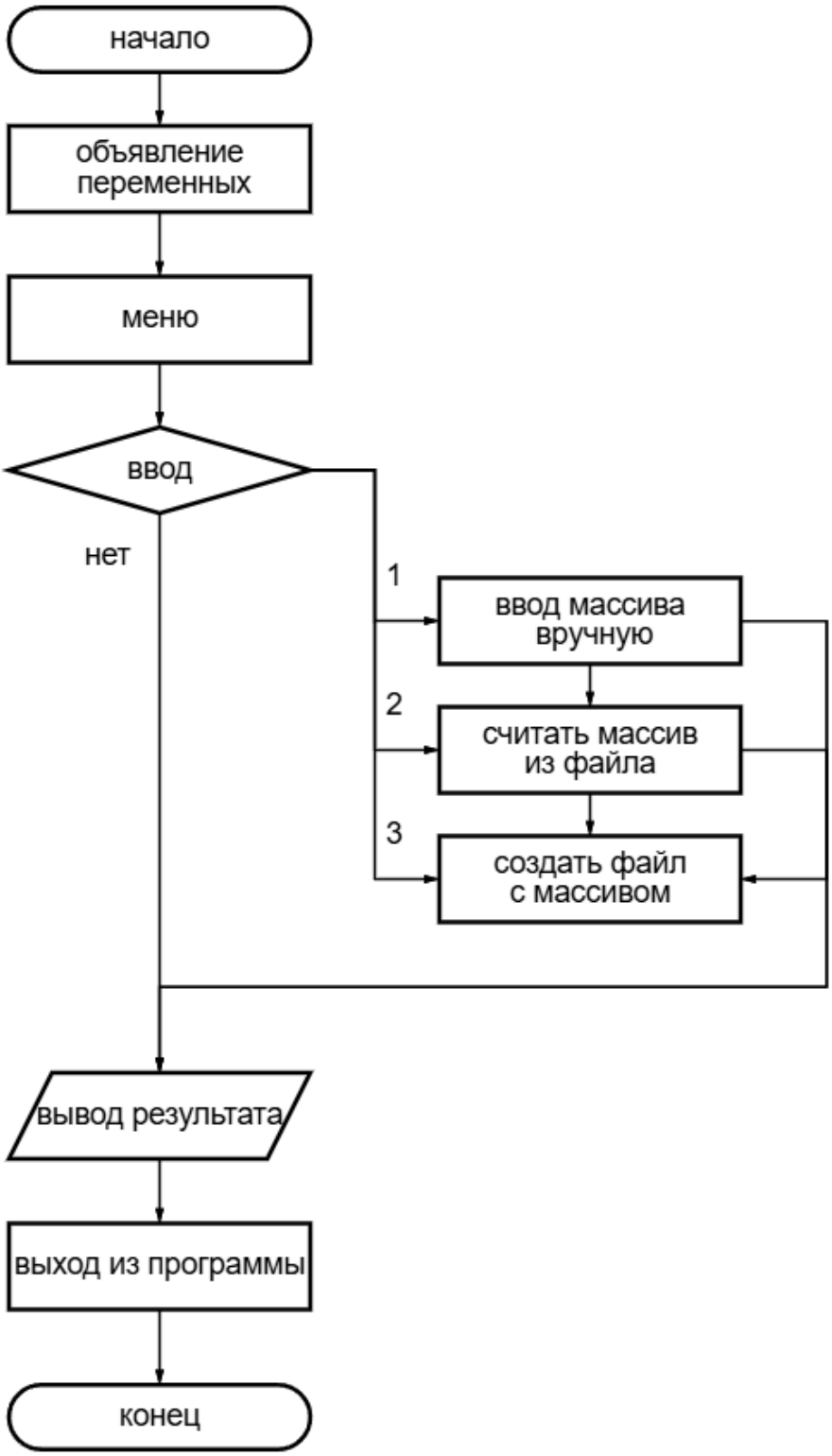

Рисунок 2 — Блок-схема функции *main*

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.

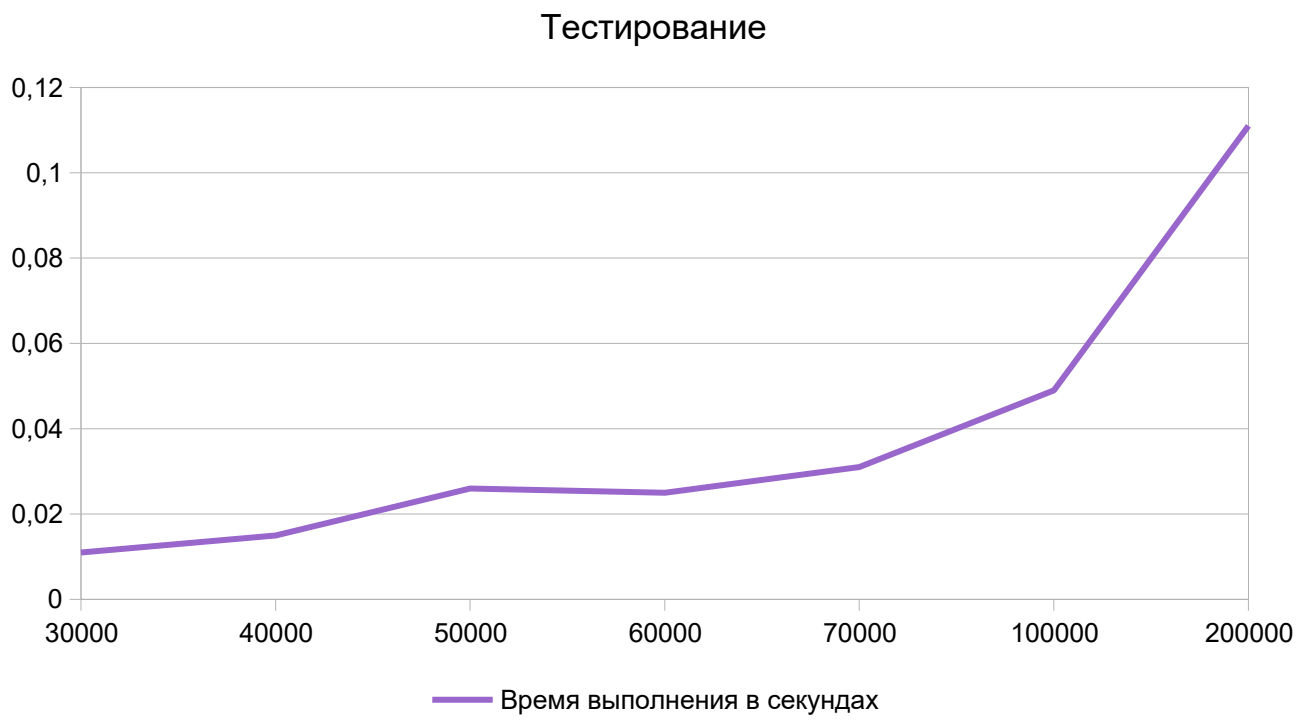

Рисунок 3 — Результаты тестирования

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 остановки 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.

7 Совместная разработка

Для удобства совместной разработки было использовано приложение *Discord*. Во время звонка был составлены и распределены задачи между составом бригады.

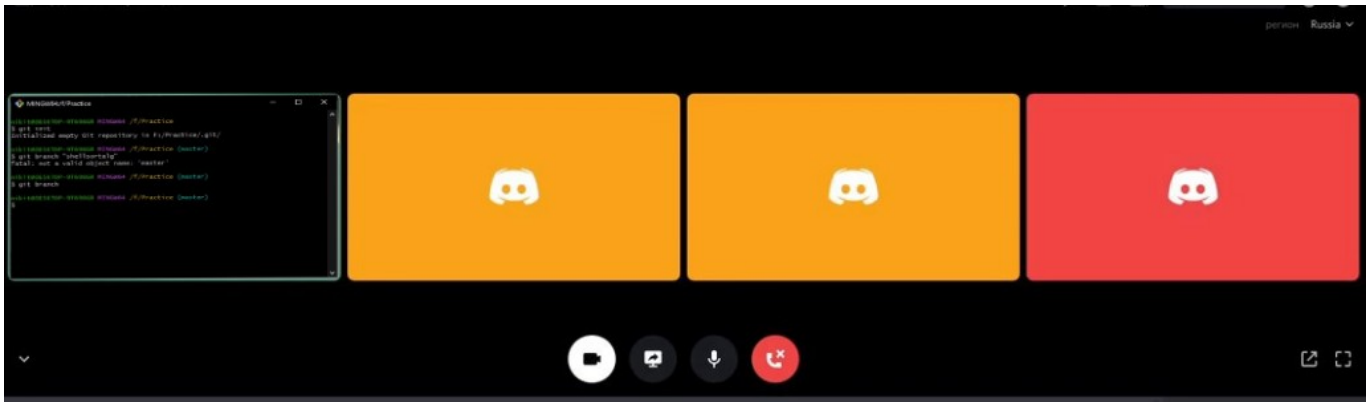

Рисунок 4 — Работа в *Discord*

Во время работы над практикой наша бригада осуществляла совместную работу в *GitHub*. Мной было написана функция случайного заполнения массива, код программы был загружен на удаленный репозиторий *Github*, сначала на ветку *shell-algsort*, а потом на ветку *main*.

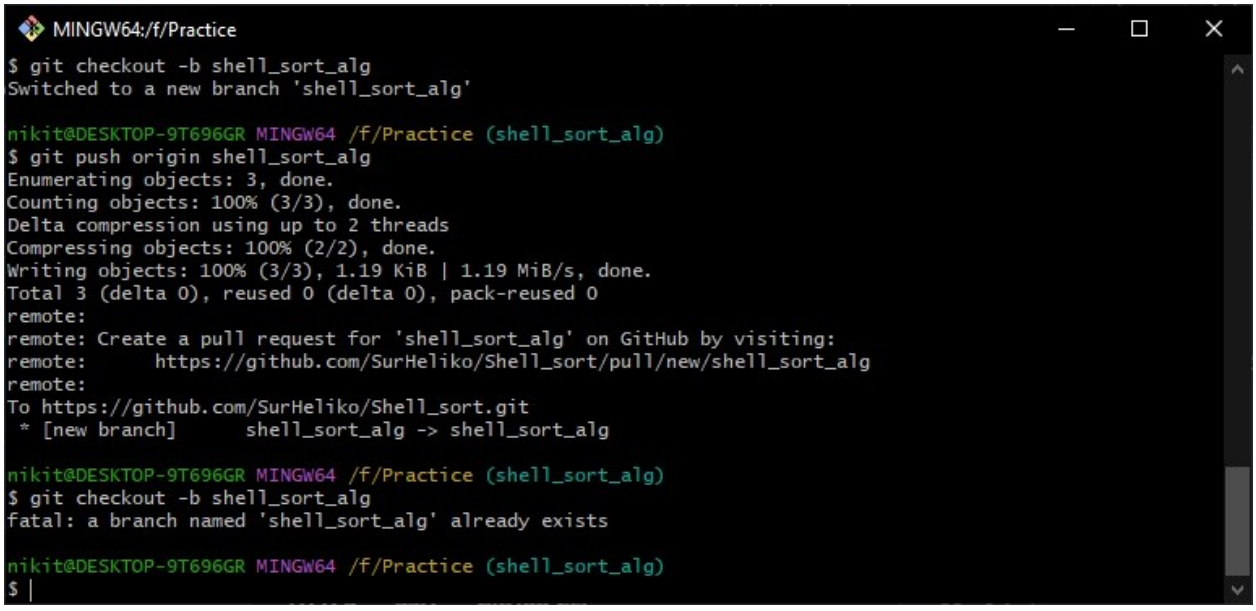

Рисунок 5 — Загрузка алгоритма на *GitHub*

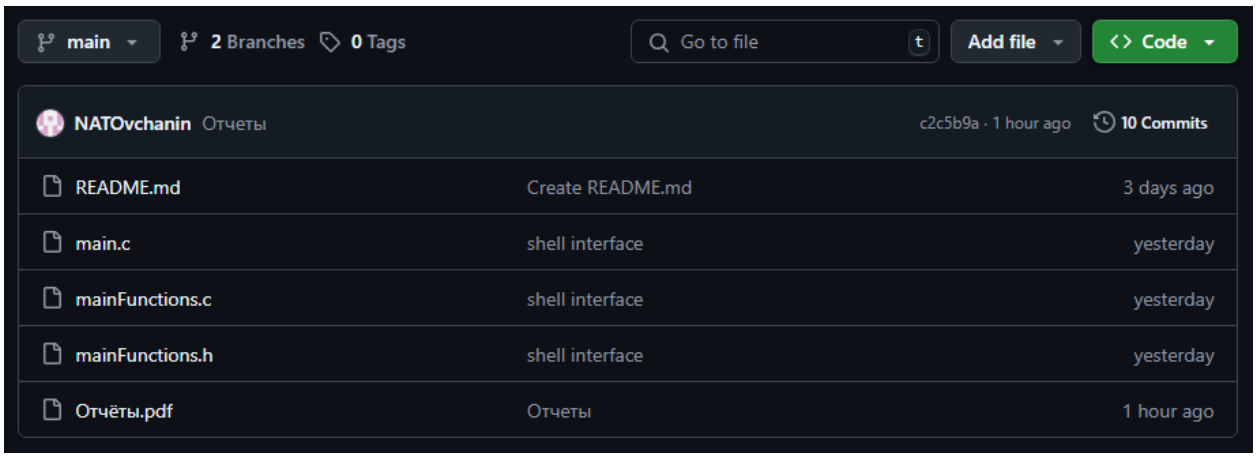

Рисунок 6 — Ветка *main*

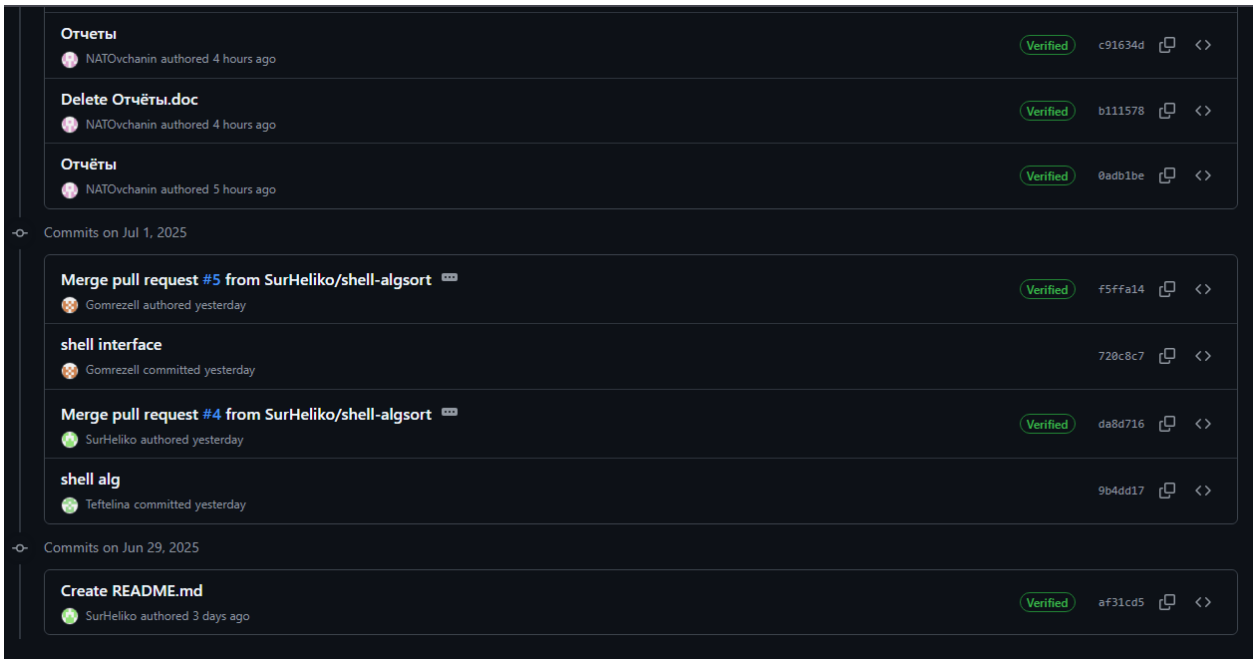

Рисунок 7 — История коммитов

Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервисов *GitHub* и *Discord*, навыки использования программы *GitBash*. Был изучен алгоритм сортировки Шелла.

Мной был написан алгоритм, который реализует метод заполнения массива чисел случайными значениями для последующей сортировки

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

В дальнейшем программу можно улучшить путем добавления графического интерфейса.

Список используемой литературы

1. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. - Введ. 1991-01-01. - М.: Стандартинформ, 2006. - 64 с.

2. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2- е издание.: Пер. с англ. – М.,2009.

3. Shell Sort Algorithm [Электронный ресурс] // GeeksforGeeks. - URL: https://www.geeksforgeeks.org/shell-sort/ (дата обращения: 28.06.2025).

Приложение А. Листинг программы

Файл main.c:

```
#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC
#include "mainFunctions.h"
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <Windows.h>
#include <locale.h>
int main() {
    setlocale(LC_ALL, "Russian");
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

    if (GetConsoleOutputCP() != CP_UTF8) {
        printf("Warning: UTF-8 not supported!\n");
    }
    printf("Выберите действие:\n");
    printf("1: Ввод массива вручную.\n");
    printf("2: Считать массив из файла input.txt.\n");
    printf("3: Создать файл с массивом случайных чисел.
\n");
    char operation = getchar();
    getchar();
    switch (operation)
    {
    case '1': {
        printf("Введите массив чисел через пробел:\n");
        int currentCount = 0, sizeArray = 10;
                int   *arrayNum   =   malloc(sizeArray   *
sizeof(int));
        char c;
        while (scanf("%d%c", &arrayNum[currentCount++],
&c) == 2) {
            if (currentCount >= sizeArray) {
                sizeArray *= 2;
```

```
                arrayNum = realloc(arrayNum, sizeArray
* sizeof(int));
            }
            if (c == '\n') break;
        }
        arrayNum = realloc(arrayNum, --currentCount *
sizeof(int));
        time_t startTime = clock();
        shellSort(arrayNum, currentCount);
        time_t endTime = clock();

        outputFile(arrayNum, currentCount);
        printf("Отсортированный массив сохранён в файл
output.txt");

               double totalTime = (double)(endTime -
startTime) / CLOCKS_PER_SEC;
                printf("\nВремя   сортировки:   %.3f\n",
totalTime);

        free(arrayNum);
        break;
    }

    case '2': {
        int sizeArr = 0;
        int* arrayNum = readFile(&sizeArr);
        if (arrayNum != NULL) {
            time_t startTime = clock();
            shellSort(arrayNum, sizeArr);
            time_t endTime = clock();

            outputFile(arrayNum, sizeArr);
            printf("Отсортированный массив сохранён в
файл output.txt");

             double totalTime = (double)(endTime -
startTime) / CLOCKS_PER_SEC;
                      printf("\nВремя сортировки: %.3f\n",
totalTime);
```

```
        }
        else printf("Ошибка чтения файла.");
        break;
    }
    case '3': {
        int sizeArr;
        printf("Введите размер желаемого массива:\n");
        scanf("%d", &sizeArr);
        createFile(sizeArr);
        int* arrayNum = readFile(&sizeArr);
        if (arrayNum != NULL) {
            time_t startTime = clock();
            shellSort(arrayNum, sizeArr);
            time_t endTime = clock();
            outputFile(arrayNum, sizeArr);
            printf("Сгенерированный массив сохранён в
input.txt,\na отсортированный - в output.txt");
             double totalTime = (double)(endTime -
startTime) / CLOCKS_PER_SEC;
                        printf("\nВремя сортировки: %.3f\n",
totalTime);
        }
        else printf("Ошибка чтения файла.");
        break;
    }

    default:
        break;
    }

    return 0;
}
```

Файл mainFunctions.c:

```
#define _CRT_SECURE_NO_WARNINGS
#include "mainFunctions.h"



void shellSort(int* arr, int cntEl) {
    for (int step = cntEl / 2; step > 0; step /= 2)
        for (int i = step; i < cntEl; i++) {
            int temp = arr[i];
            int k;
            for (k = i; k >= step && arr[k - step] >
temp; k -= step) {
                arr[k] = arr[k - step];
            }
            arr[k] = temp;
        }
}


int createFile(int sizeArray) {
    FILE* file = fopen("input.txt", "w");
    if (file == NULL) {
        printf("\nERROR: file creation error\n");
        fclose(file);
        return 1;
    }

    srand(time(NULL));

    for (int i = 0; i < sizeArray; i++) fprintf(file,
"%d\n", rand());

    fclose(file);

    return 0;
}
```

```
int* readFile(int *sizeArr) {
    FILE* file = fopen("input.txt", "r");
    if (file == NULL) {
        printf("\nERROR: file reading error\n");
        fclose(file);
        return NULL;
    }

    int currentCount = 0;
    int sizeArray = 10;
    int* numbers = malloc(sizeArray * sizeof(int));
    int num;


    while (fscanf(file, "%d", &num) == 1) {
        if (++currentCount == sizeArray) {
            sizeArray *= 2;
             numbers = realloc(numbers, sizeArray *
sizeof(int));
        }
        numbers[currentCount - 1] = num;
    }
          numbers   =   realloc(numbers,   currentCount   *
sizeof(int));

    *sizeArr = currentCount;

    fclose(file);

    return numbers;
}


int outputFile(int* arrayNum, int sizeArray) {
    FILE* file = fopen("output.txt", "w");
    if (file == NULL) {
        fclose(file);
        return 1;
    }
```

```
    for (int i = 0; i < sizeArray; i++) fprintf(file,
"%d\n", arrayNum[i]);

    fclose(file);

    return 0;
}
```

Файл mainFunctions.h:

```
#pragma once
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

void shellSort(int* arr, int cntEl);
int createFile(int sizeArray);
int* readFile(int *sizeArr);
int outputFile(int* arrayNum, int sizeArray);
```