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)

Воробьева Маргарита Михайловна

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)

Воробьева Маргарита Михайловна

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)

Воробьева Маргарита Михайловна

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»

Воробьева М.М. 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 работу с файлами и создала меню программы. Протестировала и отладила программу. Оформила отчёт.

Бакалавр Воробьева М.М. _____ "___" _____ 2023 г.

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

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)

Воробьева Маргарита Михайловна

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 Воробьева М.М. решала следующие задачи: создание алгоритма сортировки вставками, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии сортировки вставками, реализованы метод работы с файлами и со. Во время выполнения работы Воробьева М.М. 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 Воробьева М.М. заслуживает оценки «_____».

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

Содержание

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 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);

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 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) ecs – выход.

```
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("ecs - выход\n"); //вывод пунктов меню
```

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

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

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

```
f = fopen("input.txt", "w"); открытие и создание input.txt
for (int i = 0; i < size; i++)
{
        array[i] = rand() - rand();// генерация случайных чисел
        fprintf(f, "%d  ", array[i]);// запись сгенерированного массива в файл
}
fclose(f); // закрытие файла
```

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

```
f = fopen("output.txt", "w"); //открытие и создание output.txt
```

```
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 сортировки вставками
	}
}
for (int i = 0; i < size; i++) {
	fprintf(f, "%d  ", array[i]);// запись отсортированного массива в
файл


}
fclose(f);// закрытие файла
```

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 рисунках 1, 2.

Листинг программы приведен в приложении А.

4. Схемы программы

4.1 Блок-схема программы

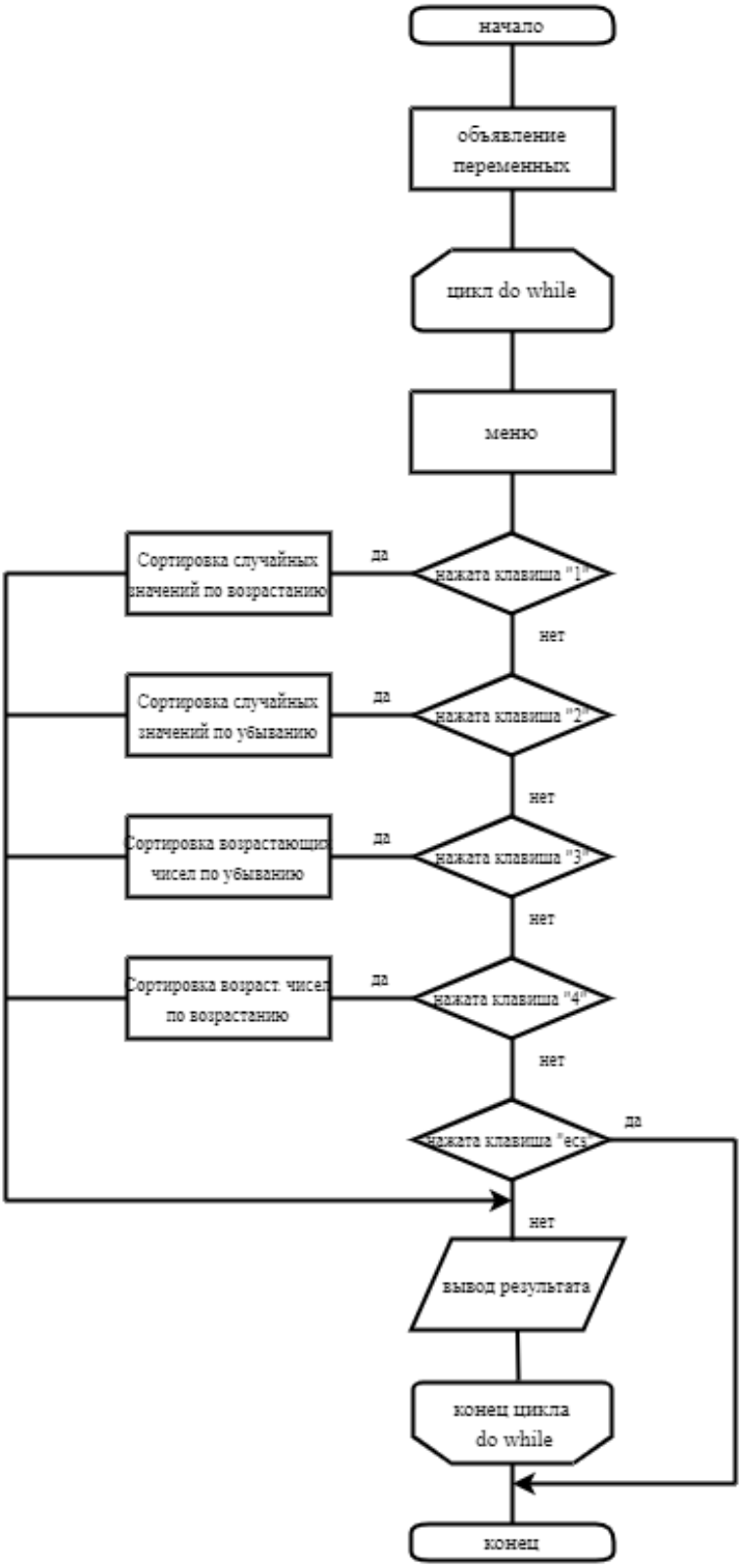

Рисунок 1 - Блок-схема программы

4.2 Блок-схема алгоритма

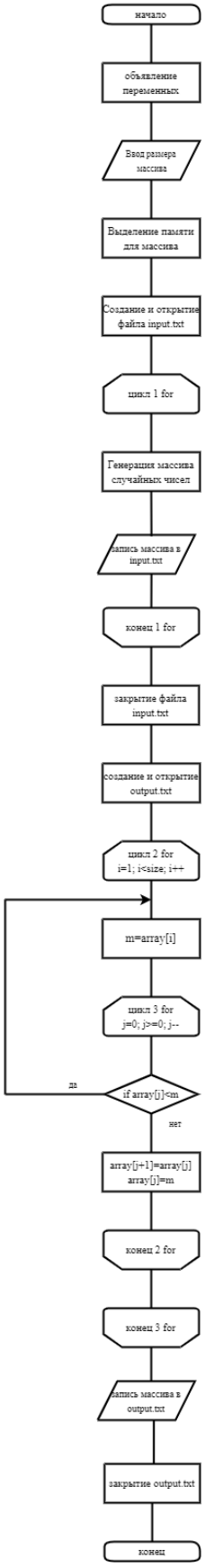

Рисунок 2 - Блок-схема алгоритма с подключенными файлами

5 Тестирование программы

Тестирование показало, что с увеличением количества элементов пропорционально увеличивается время работы программы, ниже представлен график результатов тестирования

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.

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.

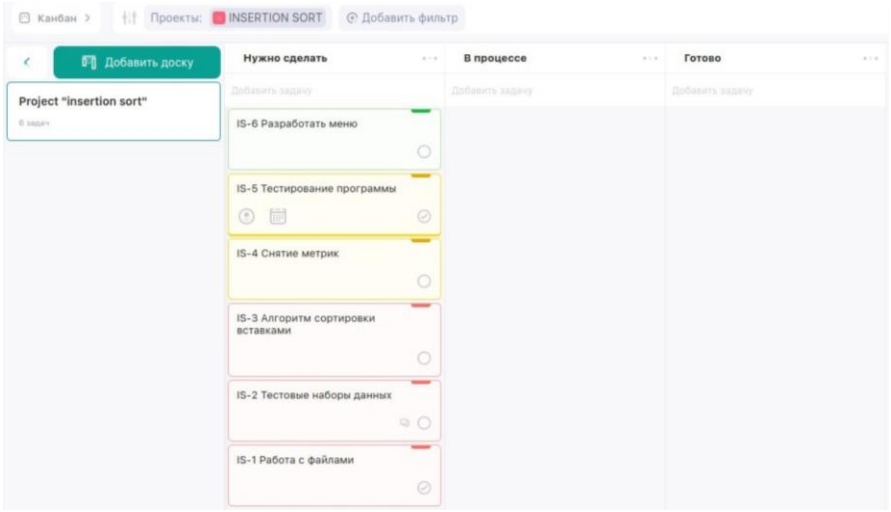

Рисунок 4 – Определение задач проекта

Разделили роли, назначили исполнителей задачам.

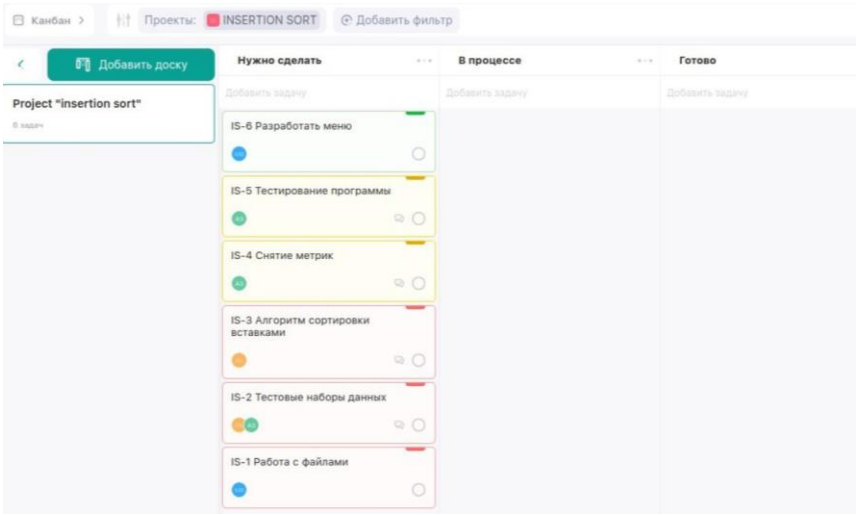

Рисунок 5 – Распределение задач проекта

Обсуждали выполнение задачи на канбан-доске.

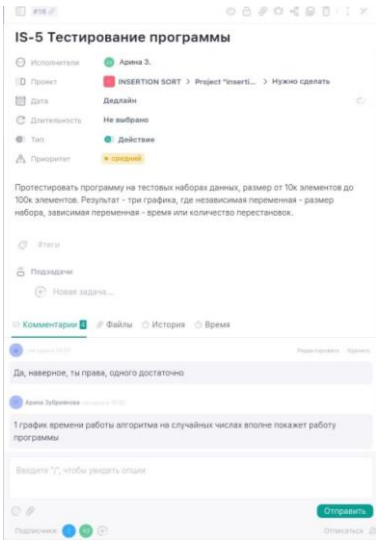

Рисунок 6 – Обсуждение задач проекта

Корректировали статус задач по мере выполнения.

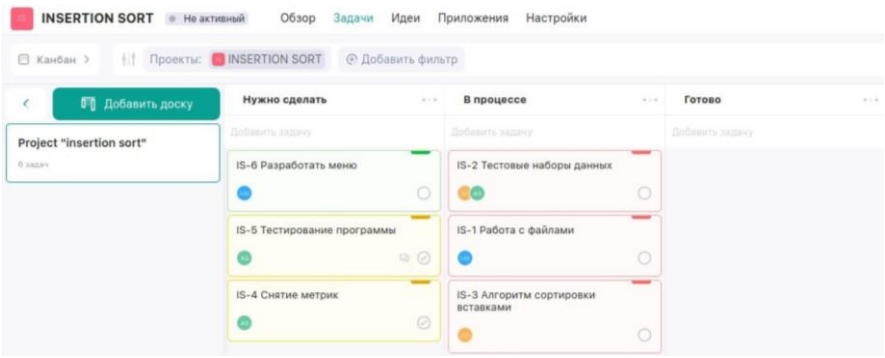

Рисунок 7 – Корректирование задач

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

Мною были подключены файлы для записи сгенерированных и отсортированных массивов, так же мною было написано меню, которое помогает ориентироваться в программе, это было зафиксировано и загружено на удаленный репозиторий Github, на ветку main.

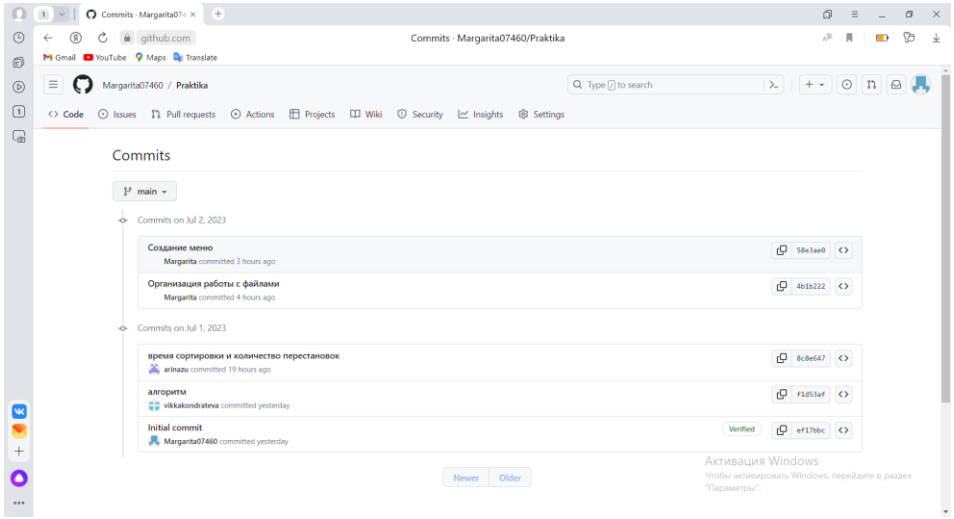

Рисунок 8 – Созданные коммиты

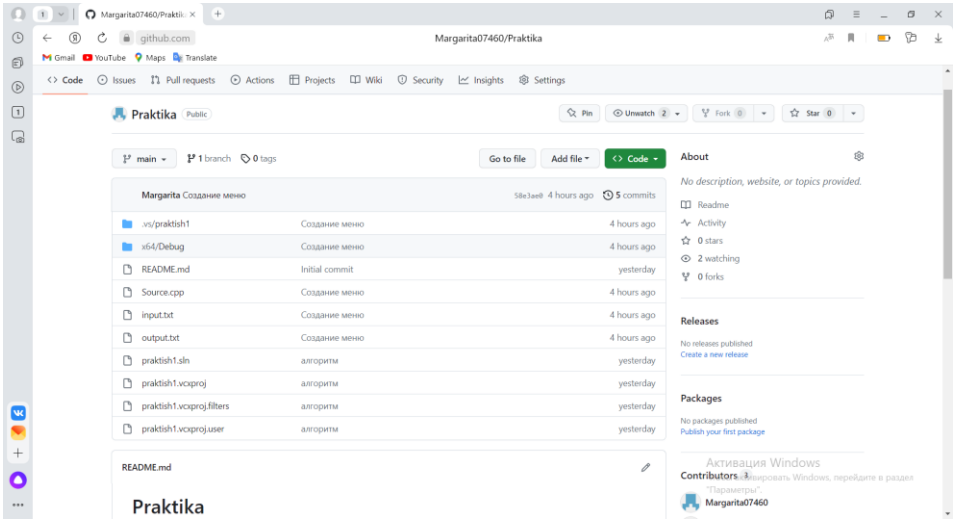

Рисунок 9– Ветка main

Для загрузки данных на локальный репозиторий, а также отправки данных на удаленный репозиторий были использованы основные команды git bash

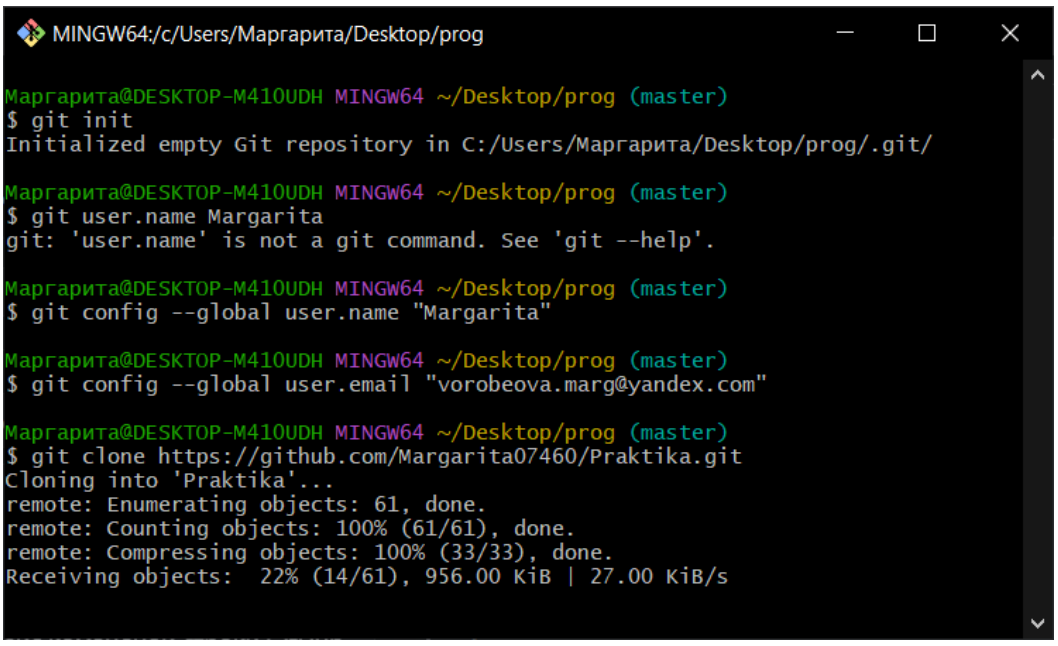

Рисунок 10 – Использование команды git clone

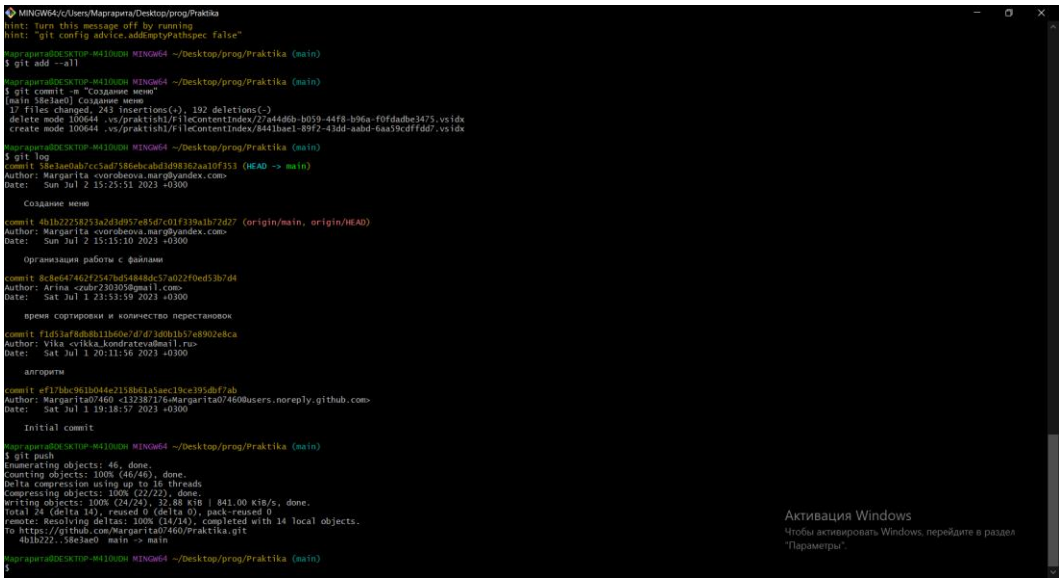

Рисунок 11 – использование команды git push и git commit

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 было создано меню программы, позволяющее выбрать одну из нескольких опций программы. Также я осуществила подключение файлов к программе, в которые выводятся результаты сгенерированного и отсортированного массивов.

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 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 C, 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. 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("ecs - 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);
}
```