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**

(2023/2024 учебный год)

                                  \_\_\_\_ Родионов Алексей Александрович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, 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 |
|  |  |

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                  Родионов Алексей Александрович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения                  1                 семестр                 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, 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 | 2 | 25.06.2024 -  25.06.2024 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 26.06.2024 –  28.06.24 |  |
| 3 | Разработка алгоритма | 43 | 01.07.24 –  03.07.24 |  |
| 4 | Описание алгоритма и программы | 18 | 03.07.24 –  04.07.24 |  |
| 5 | Тестирование | 5 | 04.07.24 –  05.07.24 |  |
| 6 | Получение и анализ результатов | 10 | 05.07.24 –  08.07.24 |  |
| 7 | Оформление отчёта | 15 | 05.07.24 –  08.07.2024 |  |
|  | **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**

(2023/2024 учебный год)

                                  Родионов Алексей Александрович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Родионов А.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. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Родионов А.А. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель Карамышева Н.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 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)

                                            Прожорин Егор Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

В процессе выполнения практики Родионов А.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. Во время выполнения работы Родионов А.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 Прожорин Е. А. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики к.т.н., доцент, Карамышева Н.С. « » 2024 г.

**Содержание**

[**Введение** 3](#_Toc171040964)

[**1 Постановка задачи** 4](#_Toc171040965)

[**1.1 Плюсы алгоритма Шейкер-сортировки:** 4](#_Toc171040966)

[**1.2 Минусы алгоритма сортировки Шелла:** 5](#_Toc171040967)

[**1.3 Типичные сценарии применения данного алгоритма:** 5](#_Toc171040968)

[**3 Описание программы** 5](#_Toc171040969)

[**4 Схемы программы** 8](#_Toc171040970)

[**4.1 Блок-схема программы** 8](#_Toc171040971)

[**4.2 Блок-схема алгоритма** 9](#_Toc171040972)

[**5 Тестирование программы** 9](#_Toc171040973)

[**6 Отладка** 11](#_Toc171040974)

[**7 Совместная работа** 13](#_Toc171040975)

[**Заключение** 15](#_Toc171040976)

[**Список используемых источников** 16](#_Toc171040977)

[**Приложение А. Листинг программы** 17](#_Toc171040978)

# **Введение**

В современном мире информационных технологий эффективная обработка данных играет ключевую роль в различных сферах деятельности. Одной из фундаментальных задач в области программирования является сортировка данных, которая позволяет упорядочивать информацию для дальнейшего анализа и использования. Задачи по сортировке данных часто встречаются в различных профессиональных областях. Алгоритмы сортировки составляют отдельный класс алгоритмов, которые используются почти во всех задачах обработки информации. Они тесно связаны друг с другом и применяются для обеспечения более быстрого поиска.

Кроме того, алгоритмы сортировки играют ключевую роль в решении многих задач, таких как поиск, фильтрация, группировка, аналитика данных и т.д. Они помогают упорядочивать данные таким образом, чтобы было удобно работать с ними и извлекать из них нужную информацию.

Таким образом, понимание и применение алгоритмов сортировки является необходимым навыком для специалистов, занимающихся обработкой и анализом данных, и позволяет им эффективно управлять информационными процессами.

Метод Шелла, или сортировка Шелла в структуре данных, представляет собой эффективный алгоритм сортировки сравнением на месте. Он назван в честь Дональда Шелла, когда он предложил первоначальную идею еще в 1959 году. Сортировка Шелла — это обобщенное расширение алгоритма сортировки вставками. Фундаментальная идея этого алгоритма сортировки состоит в том, чтобы сгруппировать элементы, находящиеся далеко друг от друга, и соответствующим образом отсортировать их. Затем постепенно уменьшается промежуток между ними. Сортировка Шелла превосходит среднее время выполнения сортировки вставками путем сравнения и обмена элементами, которые находятся далеко.

# 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 Шелла над данными, находящимися в массиве, записать отсортированные данные в другой файл и посчитать время выполнения сортировки массива.

Использовать сервис GitHub для совместной работы. Создать и выложить коммиты, характеризующие действия, выполненные каждым участником бригады.

Оформить отчет по проведенной практике.

## **1.1 Плюсы алгоритма Шейкер-сортировки:**

Сортировка Шелла работает быстрее, чем многие другие алгоритмы сортировки для небольших массивов. Это достигается за счет предварительного частичного упорядочивания элементов с использованием большого шага, который постепенно уменьшается.

Алгоритм достаточно прост для понимания и реализации. Он не требует большого объема дополнительной памяти, что делает его удобным для использования в ограниченных средах.

Сортировка Шелла может использоваться для сортировки массивов любых размеров, в отличие от некоторых специализированных алгоритмов сортировки. Она может применяться для сортировки данных в массивах, списках и других структурах данных.

## **1.2 Минусы алгоритма сортировки Шелла:**

Хотя сортировка Шелла работает быстрее, чем простые алгоритмы, такие как пузырьковая сортировка, она не так эффективна для больших массивов по сравнению с более сложными алгоритмами, такими как быстрая сортировка или сортировка слиянием.

Эффективность сортировки Шелла сильно зависит от выбранной последовательности шагов. Неправильный выбор шагов может значительно снизить производительность алгоритма.

Теоретическое время выполнения алгоритма трудно предсказать и анализировать. Хотя на практике он часто показывает хорошие результаты, его худший случай сложно оценить, и он может быть медленным.

## **1.3 Типичные сценарии применения данного алгоритма:**

В интернет-магазинах для сортировки товаров по цене, популярности, рейтингу.

В играх для упорядочивания рейтингов игроков, списков достижений или инвентаря по различным критериям.

В системах с ограниченными вычислительными ресурсами, таких как микроконтроллеры и другие встраиваемые системы, сортировка Шелла может использоваться для упорядочивания данных в памяти.

# **3 Описание программы**

При запуске программы открывается окно, в котором предлагается ввести размерность массива и диапазон значений его элементов. После того, как пользователь введёт указанные данные и нажмёт кнопку «Сортировать» массив будет сгенерирован и отсортирован. Информация о том, в каких файлах находятся сгенерированный и отсортированный массивы, а также о том, сколько времени заняла сортировка, будет отображена в сообщении, которое появится после того, как программа завершит свою работу.

Перед сортировкой происходит выделение памяти для массива. Это необходимо если мы хотим задать не определенное, а произвольное количество элементов для сортировки.

int\* createArray(int size) {

return (int\*)malloc(size \* sizeof(int));

}

Далее происходит сортировка массива алгоритмом Шелла, при котором массив делится на подмассивы, основываясь на значении шага равного половине длинны массива. На каждом шаге подмассивы сортируются сортировкой ставками. Сортировка вставками сортирует массив, проходя через него один раз и перемещая каждый элемент в его правильное место по сравнению с уже отсортированными элементами. Затем размер шага уменьшается пока он не станет равным единице. На последнем этапе массив сортируется окончательно сортировкой вставками.

void shellSort(int size, int\* arrayPointer) {

int step, i, j, temp;

for (step = size / 2; step > 0; step /= 2) {

for (i = step; i < size; i++) {

temp = arrayPointer[i];

for (j = i; j >= step; j -= step) {

if (temp < arrayPointer[j - step]) arrayPointer[j] = arrayPointer[j - step];

else break;

}

arrayPointer[j] = temp;

}

}

}

void freeArray(int\* arrayPointer) {

free(arrayPointer);

}

После записи отсортированного массива в файл происходит очистка массива из оперативной памяти. Если не освободить память выделенного массива, то это может привести к утечкам памяти — ситуации, когда доступная память уменьшается с каждым запуском программы, пока она не перестанет корректно работать из-за нехватки памяти.

void freeArray(int\* arrayPointer) {

free(arrayPointer);

}

# 

# **4 Схемы программы**

## **4.1 Блок-схема программы**

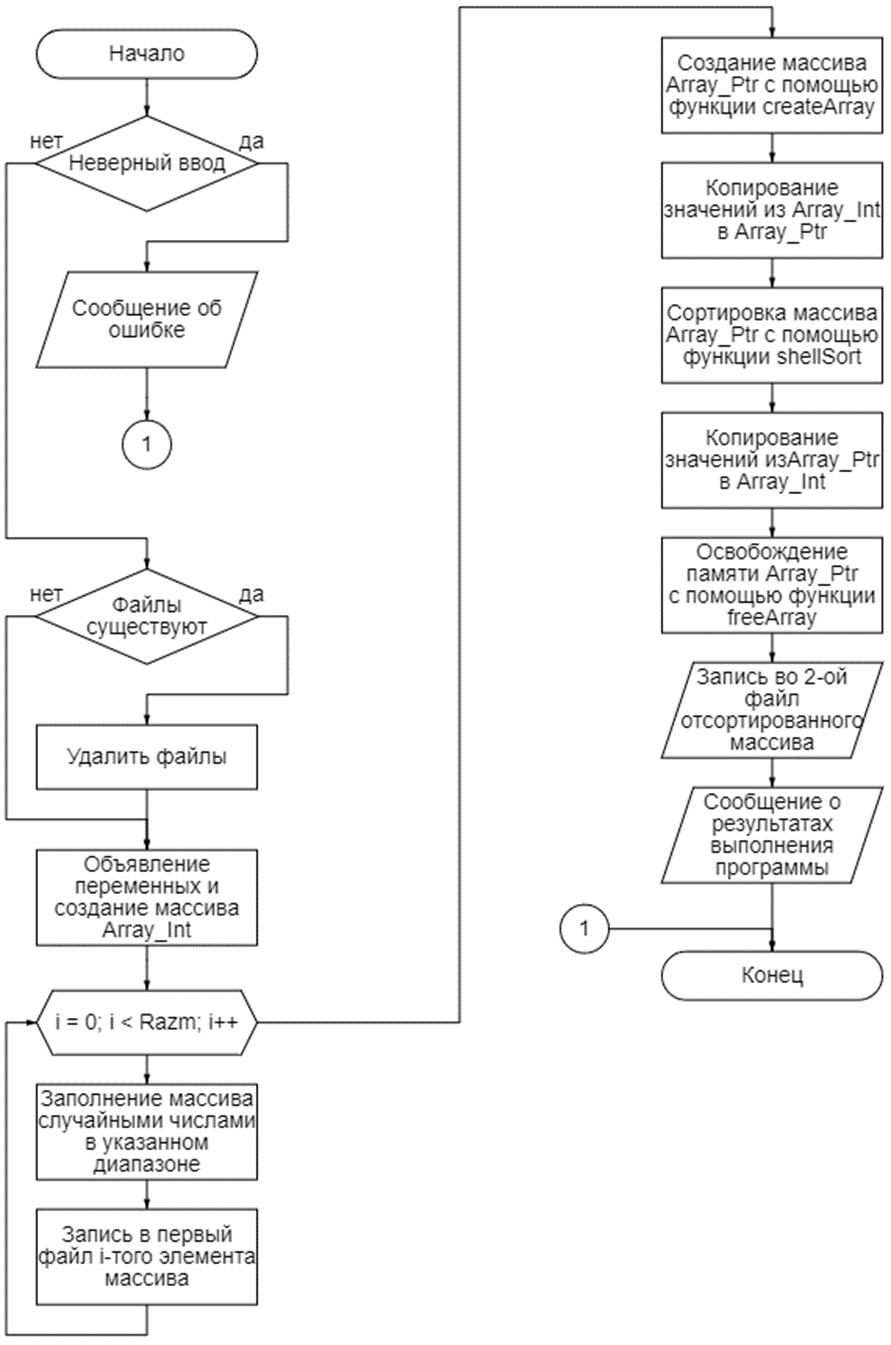


Рисунок №1 – Блок-схема программы

## **4.2 Блок-схема алгоритма**

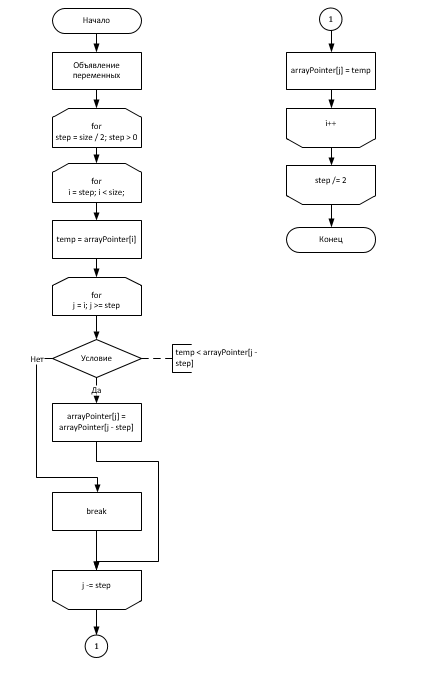


Рисунок №2 – Блок-схема алгоритма

# **5 Тестирование программы**

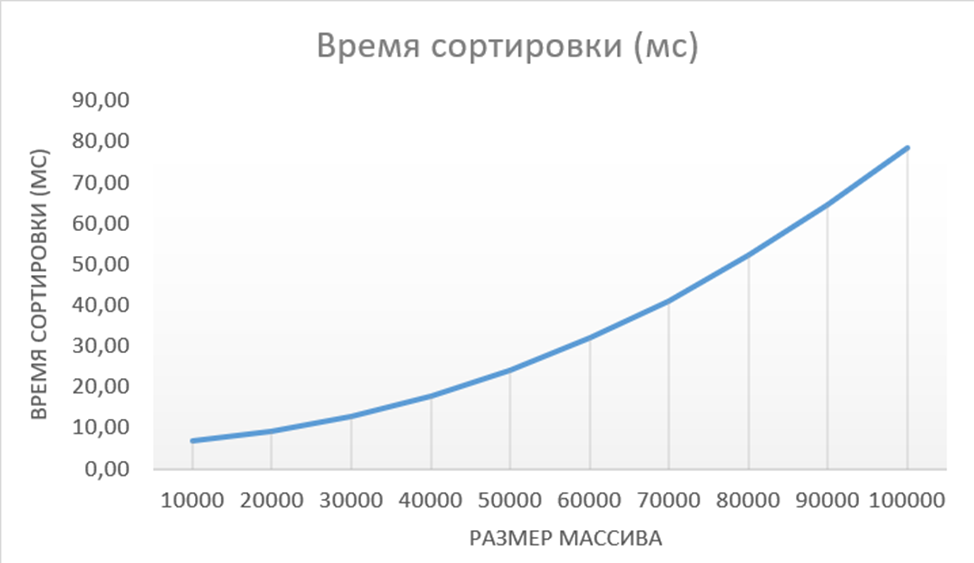
На основании анализа данных, полученных в результате тестирования алгоритма сортировки Шелла (рисунок 3), можно сделать вывод, что продолжительность выполнения программы растёт экспоненциально. Другими словами, Время сортировки растет медленно при малых значениях размера массива. По мере увеличения размера массива время сортировки растет всё быстрее. Кривая на графике становится всё более крутой, что указывает на ускоряющееся возрастание времени сортировки. Таким образом, график демонстрирует квадратичную зависимость времени сортировки от размера массива.

Рисунок №3 – Результаты тестирования

# **6 Отладка**

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных. Рассмотрим каждый из этих инструментов и способы их использования.

**Точки останова (Breakpoints)**

**Точка останова** — это инструмент, позволяющий приостановить выполнение программы на определенной строке кода. Это дает возможность изучить состояние переменных и поток выполнения программы.

**Как установить точку останова:**

1. Откройте файл с исходным кодом.
2. Найдите строку, на которой вы хотите приостановить выполнение программы.
3. Щелкните левой кнопкой мыши в области отступа слева от этой строки или нажмите F9, чтобы установить/снять точку останова.

**Типы точек останова:**

* **Безусловные:** Останавливают программу всегда, когда выполняется указанная строка.
* **Условные:** Останавливают выполнение программы, только если выполнено определенное условие.
  + Щелкните правой кнопкой мыши по точке останова и выберите **Conditions**.
  + Установите условие, например x > 10.

**Трассировка (Debugging/Stepping)**

**Трассировка** позволяет пошагово выполнять код, чтобы отслеживать поведение программы и состояние переменных.

**Основные команды трассировки:**

* **F10 (Step Over):** Выполняет текущую строку и переходит к следующей, не входя внутрь вызова функции.
* **F11 (Step Into):** Входит внутрь вызываемой функции, если это возможно.
* **Shift + F11 (Step Out):** Выходит из текущей функции и возвращается к месту вызова.
* **F5 (Continue):** Продолжает выполнение программы до следующей точки останова.

**Анализ содержимого переменных**

Во время трассировки вы можете изучить значения переменных и объектов.

**Основные способы просмотра переменных:**

1. **Окно локальных переменных (Locals):**
   * Вменювыберите **Debug** > **Windows** > **Locals**.
   * Отображает все локальные переменные в текущем контексте.
2. **Окно автопеременных (Autos):**
   * Вменювыберите **Debug** > **Windows** > **Autos**.
   * Показывает переменные, связанные с текущей и предыдущей строкой кода.
3. **Окно наблюдения (Watch):**
   * В меню выберите **Debug** > **Windows** > **Watch** > **Watch 1** (или другие).
   * Позволяет вручную добавлять интересующие переменные и следить за их значениями.
4. **Выражения в строке кода (DataTip):**
   * Наведите курсор на переменную в коде во время трассировки, чтобы увидеть ее значение.
5. **Окно быстрого просмотра (QuickWatch):**
   * Выделите интересующую переменную и нажмите Shift + F9, чтобы открыть окно быстрого просмотра.

# **7 Совместная работа**

Во время работы над данной практикой нашей бригадой осуществлялась совместную работу в GitHub.

Была написана программа сортировки, которая была загружена на удаленный репозиторий Github, на ветку main.

****

Рисунок №4 – Получение файлов с репозитория на Github

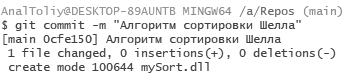


Рисунок №5 – Совершение коммита

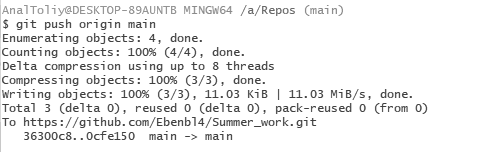


Рисунок №6 – Загрузка файлов на удаленный репозиторий Github

Для удобства совместной работы использовался мессенджер Discord. Он позволяет общаться как в голосовом, так и в текстовом чате, а также пересылать файлы. Это помогает координировать действия всей бригады для более эффективной работы над проектом.



Рисунок № 7 – окно Discord

Ссылка на удаленный репозиторий:

https://github.com/Ebenbl4/Summer\_work.git

**Заключение**

В ходе выполнения практической работы были получены навыки совместной работы с помощью сервиса Github, навыки использования Git Bash. Был реализован алгоритм сортировки Шелла. Был изучен алгоритм сортировки вставками.

Мною был написан алгоритм, осуществляющий сортировку Шелла, оформлен отчет по практике.

При выполнении практической работы были улучшены навыки программирования на языке C, а также навыки отладки и тестирования программ.

В дальнейшем программу можно улучшить, добавив функцию ручного ввода массива. Отображения количества перестановок при сортировке.

# **Список используемых источников**

1. В.В. Подбельский, С.С.Фомин. Программирование на языке Си. М.:

ФиС, 1999, 600 с.2. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.

3. Сортировка Шелла [Электронный ресурс] – URL: https://ru.wikipedia.org/ (дата обращения: 03.07.2024 г)

# **Приложение А. Листинг программы**

Файл Form1.cs

using System;

using System.Diagnostics;

using System.IO;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace Practica1

{

public partial class Form1 : Form

{

int Razm, MinVol, MaxVol;

const string File1Path = "Unsorted massive.txt";

const string File2Path = "Sorted massive.txt";

FileInfo File1Info = new FileInfo(File1Path);

FileInfo File2Info = new FileInfo(File2Path);

Stopwatch Sort\_Time = new Stopwatch();

public Form1()

{

InitializeComponent();

}

[DllImport("mySort.dll", CallingConvention = CallingConvention.StdCall)]

public static extern unsafe void shellSort(int size, int\* arrayPointer);

[DllImport("mySort.dll", CallingConvention = CallingConvention.StdCall)]

public static extern unsafe int\* createArray(int size);

[DllImport("mySort.dll", CallingConvention = CallingConvention.StdCall)]

public static extern unsafe void freeArray(int\* arrayPointer);

private void Write\_Generated\_Massive(string number)

{

try

{

using (StreamWriter writer = new StreamWriter(File1Path, true))

{

writer.WriteLine($"{number}");

}

}

catch (IOException ex)

{

MessageBox.Show($"Ошибка при записи в файл: {ex.Message}");

}

}

private void Write\_Sorted\_Massive(string number)

{

try

{

using (StreamWriter writer = new StreamWriter(File2Path, true))

{

writer.WriteLine($"{number}");

}

}

catch (IOException ex)

{

MessageBox.Show($"Ошибка при записи в файл: {ex.Message}");

}

}

private unsafe void ButtonSort\_Click(object sender, EventArgs e)

{

if ((TextBoxRazm.Text == "") || (TextBoxMinVol.Text == "") || (TextBoxMaxVol.Text == ""))

{

MessageBox.Show("Поля должны быть заполнены!");

}

else

{

if (File1Info.Exists) { File1Info.Delete(); }

if (File2Info.Exists) { File2Info.Delete(); }

Razm = Convert.ToInt32(TextBoxRazm.Text);

MinVol = Convert.ToInt32(TextBoxMinVol.Text);

MaxVol = Convert.ToInt32(TextBoxMaxVol.Text);

try

{

Random rand = new Random();

int[] Array\_Int = new int[Razm];

for (int i = 0; i < Razm; i++)

{

Array\_Int[i] = rand.Next(MinVol, MaxVol);

Write\_Generated\_Massive($"{Array\_Int[i]}");

}

Sort\_Time.Start();

IntPtr Array\_Ptr = (IntPtr)createArray(Razm);

Marshal.Copy(Array\_Int, 0, Array\_Ptr, Razm);

shellSort(Razm, (int\*)Array\_Ptr);

Marshal.Copy(Array\_Ptr, Array\_Int, 0, Razm);

freeArray((int\*)Array\_Ptr);

Sort\_Time.Stop();

string Array\_Str = string.Join("\n", Array\_Int);

Write\_Sorted\_Massive(Array\_Str);

string elapsedSeconds = Sort\_Time.Elapsed.TotalSeconds.ToString("F5");

MessageBox.Show($"Массив создан и отсортирован.\nСгенерированный массив находится в файле \"{File1Path}\".\nОтсортированный массив находится в файле \"{File2Path}\".\nВремя сортировки: {elapsedSeconds}c.");

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}");

}

}

}

private void TextBoxRazm\_TextChanged(object sender, EventArgs e)

{

}

private void TextBoxMinVol\_TextChanged(object sender, EventArgs e)

{

}

private void TextBoxMaxVol\_TextChanged(object sender, EventArgs e)

{

}

}

}

Файл mySort.dll

#include "pch.h"

#include "mySort.h"

#include <malloc.h>

void shellSort(int size, int\* arrayPointer) {

int step, i, j, temp;

for (step = size / 2; step > 0; step /= 2) {

for (i = step; i < size; i++) {

temp = arrayPointer[i];

for (j = i; j >= step; j -= step) {

if (temp < arrayPointer[j - step]) arrayPointer[j] = arrayPointer[j - step];

else break;

}

arrayPointer[j] = temp;

}

}

}

int\* createArray(int size) {

return (int\*)malloc(size \* sizeof(int));

}

void freeArray(int\* arrayPointer) {

free(arrayPointer);

}

**Приложение В. Окно программы**

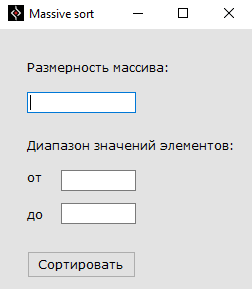


Рисунок 8 – Окно программы