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)

                                Прохоров Даниил Сергеевич

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)

                                          Прохоров Даниил Сергеевич

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)

                                     Прохоров Даниил Сергеевич

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»

Прохоров Д.С. 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. Оформил отчёт.

Бакалавр Прохоров Д.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                  Прохоров Даниил Сергеевич

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 Прохоров Д.С. 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 Прохоров Д.С. 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 Прохоров Д.С. заслуживает оценки «\_\_\_\_\_\_».

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

**Оглавление**

[Введение 1](#_Toc9173)

[1. Разработка и описание алгоритма 3](#_Toc15203)

[2. Описание программы 7](#_Toc7320)

3. Совместная работа [13](file:///C:\Users\valvi\source\repos\practice1\Отчёт%20по%20практике%20Прохоров%201курс.docx#_Toc23388)

[Заключение 11](#_Toc2943)

[Приложение А](#_Toc23388) 12

[Приложение Б 13](#_Toc23388)

#### Введение

Язык программирования C# представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C# является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык C и C++, а также в Java, от которых он унаследовал многие концепции и синтаксические особенности. В начале 2000-х годов язык C# был разработан компанией Microsoft под руководством Андерса Хейлсберга. Он был создан как часть платформы .NET и изначально был предназначен для разработки приложений под операционную систему Windows. Фактически, C# вобрал в себя лучшие черты своих предшественников и добавил ряд новых возможностей, таких как автоматическое управление памятью через сборку мусора и расширенные средства для работы с асинхронным программированием.

C# является мощным языком, предоставляя разработчикам богатый набор возможностей для создания современных приложений. Он широко используется для разработки настольных приложений, веб-приложений и веб-сервисов. Благодаря платформе .NET, C# предоставляет отличные средства для создания высокопроизводительных и масштабируемых решений, что делает его идеальным выбором для корпоративного программного обеспечения и облачных сервисов. Многие популярные системы управления контентом (CMS), такие как Umbraco и Sitecore, написаны на C#.

Однако применение данного языка не ограничивается только корпоративными приложениями. C# активно используется в разработке игр, благодаря мощному движку Unity, который является одним из лидеров на рынке игровых движков. С помощью Unity и C# можно создавать игры для различных платформ, включая мобильные устройства, консоли и ПК.

Также в последнее время C# нашел применение в мобильной разработке, особенно благодаря фреймворку Xamarin, который позволяет создавать кроссплатформенные мобильные приложения для iOS и Android на основе единого кода. Кроме того, C# используется для создания микросервисов и различных вспомогательных сервисов, которые обслуживают веб-приложения и облачные системы.

В общем, C# – это язык широкого пользования, на котором можно создавать практически любые виды программ.

#### Разработка и описание алгоритма

Сортировка Шелла (или метод Шелла) – это один из первых алгоритмов сортировки, разработанный Дональдом Шеллом в 1959 году. Этот алгоритм является улучшенной версией сортировки вставками и работает по принципу разделения и последующей сортировки подмассивов, что позволяет существенно уменьшить количество перестановок.

Он работает следующим образом:

* Изначально используется большой шаг (расстояние) для сравнения и обмена элементов.
* На каждом этапе шаг уменьшается, и элементы, отстоящие друг от друга на это расстояние, сравниваются и при необходимости меняются местами.
* В конечном итоге шаг уменьшается до 1, и происходит окончательная сортировка вставками..

Основные принципы работы алгоритма:

1. Разделение массива на подмассивы:

Алгоритм начинается с разделения исходного массива на несколько подмассивов, которые образуются путем выбора элементов через определенный интервал (gap). Например, если интервал равен 5, то подмассивы будут состоять из элементов, расположенных через каждые 5 позиций.

2. Сортировка подмассивов:

Каждый подмассив сортируется с помощью сортировки вставками. Поскольку эти подмассивы относительно небольшие, сортировка вставками на них работает очень эффективно.

3. Уменьшение интервала (gap):

После сортировки всех подмассивов текущего интервала, интервал уменьшается и процесс повторяется. Обычно, интервал делится на определенное число (например, на 2 или на 2.2), пока не станет равным 1.

4. Конечная сортировка:

Когда интервал становится равным 1, массив полностью сортируется сортировкой вставками. На этом этапе массив уже практически отсортирован благодаря предыдущим шагам, что делает последнюю сортировку вставками очень быстрой.

Пример работы алгоритма:

Рассмотрим массив [23, 29, 15, 19, 31, 7, 9, 5, 2].

1. Выбираем начальный интервал (gap). Пусть это будет 4.

Подмассивы: [23, 31], [29, 7], [15, 9], [19, 5], [2]

2. Сортируем каждый подмассив:

[23, 31] -> [23, 31]

[29, 7] -> [7, 29]

[15, 9] -> [9, 15]

[19, 5] -> [5, 19]

[2] -> [2]

3. Новый интервал: gap = 2.

Подмассивы: [23, 15, 31, 7, 19, 9, 5, 2]

4. Сортируем каждый подмассив:

[23, 15, 31, 7, 19, 9, 5, 2] -> [15, 23, 7, 31, 9, 19, 2, 5]

5. Новый интервал: gap = 1.

Полностью сортируем массив вставками:

[15, 7, 9, 2, 5, 19, 23, 31] -> [2, 5, 7, 9, 15, 19, 23, 31]

Сложность алгоритма:

Время выполнения сортировки Шелла зависит от выбора последовательности интервалов. В худшем случае сложность алгоритма может достигать O(n^2), однако на практике она обычно значительно ниже. Для некоторых оптимальных последовательностей интервалов (например, последовательность Хиббарда, Седжвика) сложность может приближаться к O(n log^2 n), что делает сортировку Шелла одной из самых эффективных среди простых алгоритмов сортировки.

Сортировка Шелла является хорошим выбором для сортировки небольших и средних массивов и демонстрирует превосходную производительность по сравнению с простыми методами, такими как сортировка вставками или пузырьковая сортировка.

Плюсы:

Улучшенная производительность:

Сортировка Шелла работает быстрее, чем простые алгоритмы сортировки, такие как сортировка вставками и пузырьковая сортировка, особенно для больших массивов.

Эффективность при неупорядоченных данных:

Алгоритм хорошо справляется с неупорядоченными данными, значительно уменьшая количество необходимых перестановок элементов.

Простота реализации:

Алгоритм относительно прост в реализации и понимании по сравнению с более сложными алгоритмами сортировки, такими как быстрая сортировка или пирамидальная сортировка.

Гибкость в выборе шагов:

Возможность использовать различные последовательности шагов (расстояний), что может повлиять на производительность алгоритма и адаптировать его под конкретные задачи.

Работа на месте:

Сортировка Шелла является алгоритмом сортировки на месте (in-place), не требуя дополнительной памяти для хранения промежуточных массивов.

Минусы:

Сложность в определении оптимальной последовательности шагов:

Производительность алгоритма сильно зависит от выбора последовательности шагов, и определение оптимальной последовательности может быть сложной задачей.

Менее предсказуемое время выполнения:

Время выполнения алгоритма может варьироваться в зависимости от начальной последовательности данных и выбранной последовательности шагов, что делает его менее предсказуемым по сравнению с алгоритмами с фиксированной сложностью.

Не самая лучшая асимптотическая сложность:

Несмотря на улучшенную производительность по сравнению с простыми алгоритмами, сортировка Шелла не может конкурировать с алгоритмами, такими как быстрая сортировка или пирамидальная сортировка, по наихудшей временной сложности. В худшем случае сложность может достигать O(n2)O(n^2)O(n2).

Отсутствие стабилности:

Сортировка Шелла не является стабильной, то есть равные элементы могут поменять порядок относительно друг друга после сортировки.

1. **Описание программы**

Обмен элементов  
Метод Swap

static void Swap(ref int a, ref int b)

{

(a, b) = (b, a);

}

Этот метод обменивает значения двух переменных. Ключевое слово ref означает, что параметры a и b передаются по ссылке, что позволяет методу изменить их значения напрямую.

(a, b) = (b, a); - Используется кортеж для одновременного обмена значений a и b.

Метод ShellSort

static int[] ShellSort(int[] array)

{

var d = array.Length / 2;

while (d >= 1)

{

for (var i = d; i < array.Length; i++)

{

var j = i;

while ((j >= d) && (array[j - d] > array[j]))

{

Swap(ref array[j], ref array[j - d]);

j = j - d;

}

}

d = d / 2;

}

return array;

}

**Инициализация шага (расстояния)** d:

var d = array.Length / 2;

Начальное значение шага d устанавливается как половина длины массива.

**Внешний цикл** while:

while (d >= 1)

Цикл продолжается, пока d не станет меньше 1. В каждой итерации значение d уменьшается вдвое.

**Цикл** for **по элементам массива**:

for (var i = d; i < array.Length; i++)

Этот цикл проходит по элементам массива, начиная с индекса d до конца массива.

**Внутренний цикл** while **для сравнения и обмена элементов**:

var j = i;

while ((j >= d) && (array[j - d] > array[j]))

{

Swap(ref array[j], ref array[j - d]);

j = j - d;

}

В этом цикле происходит сравнение элементов, отстоящих друг от друга на расстояние d. Если элемент array[j - d] больше элемента array[j], то они обмениваются местами. Это продолжается до тех пор, пока не будет достигнут предел индекса d или элементы в нужной позиции.

**Уменьшение шага** d:

d = d / 2;

После завершения внутреннего цикла шаг d уменьшается вдвое, и процесс повторяется до тех пор, пока d не станет меньше 1.

**Возврат отсортированного массива**:

return array;

Метод возвращает отсортированный массив.

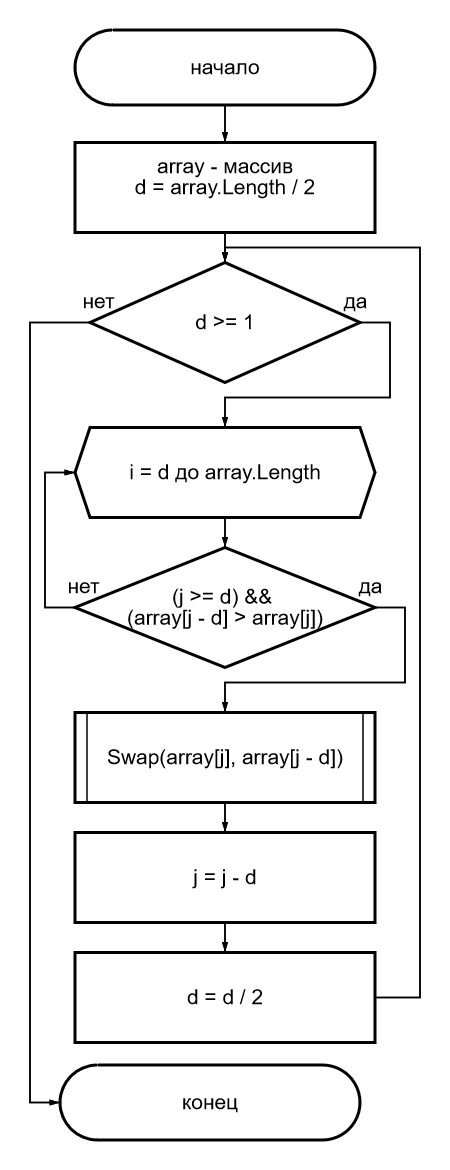


Рисунок 1 – Блок-схема алгоритма

1. **Совместная работа**

Во время работы над данной практикой, нашей бригадой осуществлялась совместная работа в GitHub.

Данная программа была написана Прохоровым Д.С и владельцем репозитория Панькиным Р.А. Молев П.О. провёл тестирование данной программы

После написания алгоритма сортировки программы Прохоровым Д.С, она была выгружена на удаленный репозиторий Github. Сделал коммит, и выгрузил обновленный код программы на удаленный репозиторий GitHub.

После этого, второй участник – Панькин Р.А. загрузил данную программу себе на компьютер, с помощью git clone <https://github.com/Eliksirowich/practice1.git> , и добавил в нее алгоритм, считающий время работы программы в секундах ,графический интерфейс и работу с файлами. Сделал коммит, и выгрузил обновленный код программы на удаленный репозиторий GitHub.

Далее, третий участник – Молев П.О загрузил программу себе на компьютер, с помощью git clone <ссылка>, и провёл тестировку на наличие ошибок в программе.

Ссылка на репозиторий: <https://github.com/Eliksirowich/practice1.git>

#### Заключение

На основании анализа данных, полученных в результате тестирования алгоритма сортировки Шелла, можно сделать следующие выводы:

Время, затраченное на выполнение программы, увеличивается нелинейно относительно количества элементов. Хотя сортировка Шелла демонстрирует улучшенную производительность по сравнению с сортировкой вставками, её эффективность зависит от начального выбора шага (расстояния между сравниваемыми элементами).

При малых размерах массивов сортировка Шелла выполняется быстрее за счет уменьшения количества необходимых обменов. Однако, с увеличением количества элементов наблюдается тенденция к субквадратичному росту времени выполнения, особенно в худшем случае.

Таким образом, можно сказать, что алгоритм сортировки Шелла не обладает строгой линейной зависимостью времени работы от количества элементов, но показывает значительные улучшения по сравнению с простыми алгоритмами сортировки для массивов среднего размера.

#### Список литературы

1. Документация С# - <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>
2. Адитья Бхаргава - Грокаем алгоритмы
3. Принцип сортировки Шелла - https://training.epam.kz/ru/blog/449

#### Приложение А

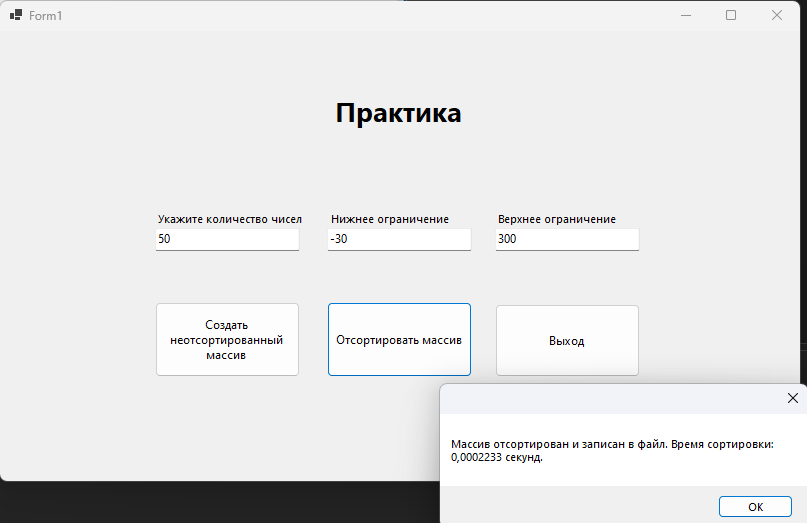


Рисунок 2 – Окно взаимодействия

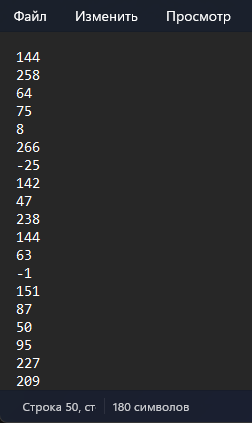


Рисунок 3 – Неотсортированный список

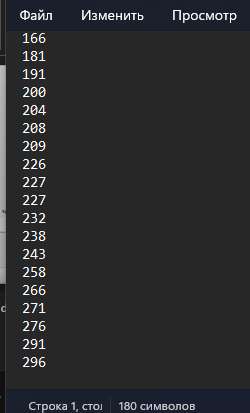


Рисунок 4 – Отсортированный список

#### Приложение Б

Листинг:

**Sort.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace practice

{

public partial class Form1

{

//метод для обмена элементов

static void Swap(ref int a, ref int b)

{

(a, b) = (b, a);

}

static int[] ShellSort(int[] array)

{

//расстояние между элементами, которые сравниваются

var d = array.Length / 2;

while (d >= 1)

{

for (var i = d; i < array.Length; i++)

{

var j = i;

while ((j >= d) && (array[j - d] > array[j]))

{

Swap(ref array[j], ref array[j - d]);

j = j - d;

}

}

d = d / 2;

}

return array;

}

}

}

Form1.cs:

using System;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using System.Diagnostics;

namespace practice

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

Random random = new Random();

int[] array = new int[1000000];

for (int i = 0; i < array.Length; i++)

{

array[i] = random.Next(-10001, 10001);

}

string filePath = "array.txt";

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (int number in array)

{

writer.WriteLine(number);

}

}

MessageBox.Show("Файл с неотсортированным массивом создан.");

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string filePath = "array.txt";

if (!File.Exists(filePath))

{

MessageBox.Show("Файл с массивом не найден.");

return;

}

string[] lines = File.ReadAllLines(filePath);

int[] array = new int[lines.Length];

for (int i = 0; i < lines.Length; i++)

{

array[i] = int.Parse(lines[i]);

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

array = ShellSort(array);

stopwatch.Stop();

string sortedFilePath = "sorted\_array.txt";

using (StreamWriter writer = new StreamWriter(sortedFilePath))

{

foreach (int number in array)

{

writer.WriteLine(number);

}

}

MessageBox.Show($"Массив отсортирован и записан в файл. Время сортировки: {stopwatch.Elapsed.TotalSeconds} секунд.");

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}