Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.03 Прикладная информатика

Профиль Разработка и внедрение прикладных информационных систем

**КУРСОВАЯ РАБОТА**

по дисциплине программирование

*(наименование дисциплины)*

на тему: Решение типовых задач основных разделов программирования. Вариант 11

*(тема курсового проекта)*

Выполнил студент Мальцев Михаил Сергеевич курса 1 группы 22-КБ-ПИ1

*(фамилия, имя, отчество)*

Допущен к защите

*(дата)*

Руководитель (нормоконтролер) работы к.т.н., доц. В.А. Мурлина

*(подпись, дата)*

Защищен Оценка

*(дата)*

Члены комиссии:                                                                 ассистент  П.А. Косолапов

                                                                   ст. преп­­. А.А. Ковтун

*(подпись, дата)*

Краснодар

2023

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.03 Прикладная информатика

Профиль Разработка и внедрение прикладных информационных систем

УТВЕРЖДАЮ

Зав. кафедрой ИСП

к.т.н., доц.                  М.В. Янаева

« 13 »       сентября          20­­22 г.

**ЗАДАНИЕ**

на курсовую работу

Студенту Мальцев Михаил Сергеевич курса 1 группы 22-КБ-ПИ1

Тема работы: «Решение типовых задач основных разделов программирования. Вариант 11 »

(утверждена указанием директора института № от\_\_\_\_\_\_\_\_\_)

План работы:

1.

2.

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем работы:

а) пояснительная записка                                                                 с.

б) иллюстративная часть                              –                                  лист(-ов)

Рекомендуемая литература:

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок выполнения: с «  13  »       сентября        по « 25 »     декабря         2022 г.

Срок защиты: «   »                              2022 г.

Дата выдачи задания: « 13  »       сентября        2022 г.

Дата сдачи работы на кафедру: « 30 »       декабря      \_  2022 г.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_к.т.н., доц. В.А. Мурлина

(*подпись)*

Задание принял студент М.С. Мальцев

**Реферат**

Пояснительная записка к курсовой работе содержит: 31 c., 15 рис., 3 источника.

VISUAL STUDIO 2022, C#, ПРОГРАММЫ, ОДНОМЕРНЫЕ МАССИВЫ, МЕТОД СОРТИРОВКИ, АЛГОРИТМ, СОРТИРОВКА, УЛУЧШЕННЫЕ МЕТОДЫ СОРТИРОВКИ.

Цель исследования улучшенных методов сортировки заключается в изучении и определении наиболее эффективных алгоритмов сортировки, которые могут быть применены для сортировки больших объемов данных в компьютерных системах, с помощью одномерных массивов.

Результат работы: получение знаний в области структур данных и их реализации, алгоритмов для решения поставленной задачи.

В данной работе представлены два улучшенных алгоритма сортировки («быстрая сортировка» и «сортировка Шела»). В коде программы реализовано сравнение эффективности алгоритмов друг с другом, а также «сортировкой пузырьком».

Объект исследования: улучшенные методы сортировки.

Предмет исследования: «сортировка Шела», «быстрая сортировка».

**Содержание**

[**Введение** 5](#_Toc131839814)

[**1 Нормативные ссылки** 6](#_Toc131839815)

[**2 Описание предметной области** 7](#_Toc131839816)

[**2.1 Общие положения** 7](#_Toc131839817)

[**2.2 Сведения из теории** 7](#_Toc131839818)

[**2.3 Постановка задачи** 9](#_Toc131839819)

[**3 Технология разработки программы** 11](#_Toc131839820)

[**3.1 Описание алгоритма решения** 11](#_Toc131839821)

[**3.2 Блок схемы алгоритма** 12](#_Toc131839822)

[**3.3 Программная реализация** 16](#_Toc131839823)

[**3.4 Результат выполнения программы** 22](#_Toc131839824)

[**3.5 Сравнительный анализ** 24](#_Toc131839825)

[**Заключение** 29](#_Toc131839826)

[**Список использованных источников** 30](#_Toc131839827)

[**Приложение А. Тест на антиплагиат** 31](#_Toc131839828)

# **Введение**

Сортировка является одной из наиболее фундаментальных операций в информатике, которая находит широкое применение в различных областях, таких как базы данных, анализ данных, компьютерное зрение, графика и другие. Существует множество алгоритмов сортировки, каждый из которых имеет свои преимущества и недостатки.

В последние годы было проведено множество исследований по улучшению существующих методов сортировки и созданию новых алгоритмов, которые могут работать более эффективно, особенно на больших объемах данных. В этом исследовании мы сосредоточимся на изучении некоторых из этих усовершенствованных методов сортировки и оценим их производительность по некоторым параметрам.

Целью данного исследования является сравнительный анализ различных методов сортировки на основе их эффективности по количеству перестановок и затраченному времени.

Первый раздел содержит ссылки на использованные при составлении пояснительной записки ГОСТы.

Каждый из разделов со второго по пятый пояснительной записки к курсовой работе содержит основные сведения по определению и использованию соответствующей конструкции языка, содержание поставленных (в соответствии с вариантом) задач, блок-схемы алгоритмов их решений, программные реализации поставленных задач в виде листингов написанных программ, скриншоты выполнения программ, представленные для доказательства работоспособности кода и корректности разработанного алгоритма решения задач. Каждый из этих разделов представляет указанные подразделы по таким конструкция языка C# как одномерные массивы, двумерные массивы, строки и файлы.

При решении задач курсовой работы были использованы следующие методы: сравнение, анализ и синтез, моделирование, алгоритмизация.

# **1 Нормативные ссылки**

В данной пояснительной записке использованы ссылки на следующие стандарты:

–ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам;

–ГОСТ Р.7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;

–ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила.

# **2 Описание предметной области**

## **2.1 Общие положения**

Целью данной работы является разработка программы для исследования улучшенных методов сортировки в среде Microsoft Visual Studio 2022, с помощью одномерных массивов.

В качестве предмета исследования были выбраны следующие алгоритмы: «сортировка Шела», «быстрая сортировка».

## **2.2 Сведения из теории**

#### **2.2.1 Улучшенные методы сортировки**

Улучшенные методы сортировки - это алгоритмы, которые улучшают производительность и эффективность сортировки по сравнению с классическими алгоритмами сортировки. Это может быть достигнуто за счет оптимизации алгоритмической логики или улучшения времени выполнения алгоритма.

Некоторые из наиболее популярных улучшенных алгоритмов сортировки включают в себя:

1. Быстрая сортировка с выбором медианного элемента (Median QuickSort) - это улучшенный алгоритм быстрой сортировки, который выбирает медианный элемент из массива в качестве опорного, что позволяет избежать худшего случая, когда выбранный опорный элемент находится на краю массива.

2. Сортировка слиянием (MergeSort) - это алгоритм сортировки, который использует метод "разделяй и властвуй" и объединяет отсортированные подмассивы в единый отсортированный массив.

3. Быстрая сортировка с случайным выбором опорного элемента (Randomized QuickSort) - это алгоритм быстрой сортировки, который выбирает случайный элемент в качестве опорного, что помогает избежать худшего случая, когда выбранный опорный элемент находится на краю массива.

4. Сортировка Хоара с использованием трех опорных элементов (Three-way QuickSort) - это улучшенный алгоритм быстрой сортировки, который использует три опорных элемента вместо одного, чтобы разделить массив на три части: элементы, меньшие первого опорного элемента, элементы, равные первому опорному элементу, и элементы, большие первого опорного элемента.

5. Сортировка Шелла с динамическим выбором шага (Dynamic ShellSort) - это улучшенный алгоритм сортировки Шелла, который динамически выбирает шаг сортировки в зависимости от размера массива, что позволяет ускорить выполнение алгоритма для больших массивов.

#### **2.2.2 Метод «быстрой сортировки»**

Быстрая сортировка (QuickSort) - это один из самых популярных алгоритмов сортировки, который использует метод "разделяй и властвуй" для упорядочивания элементов в массиве. Алгоритм быстрой сортировки работает по следующей схеме:

1. Выберите опорный элемент из массива. Обычно это первый, последний или случайный элемент в массиве.

2. Разделите массив на две подгруппы, одну содержащую элементы, меньшие или равные опорному, и другую содержащую элементы, большие опорного.

3. Рекурсивно примените быструю сортировку к каждой из подгрупп.

4. Объедините отсортированные подгруппы в один отсортированный массив.

Шаг разделения массива на две подгруппы выполняется с помощью двух указателей, которые начинаются соответственно с первого и последнего элемента в подмассиве. Затем оба указателя перемещаются внутрь подмассива, пока они не встретятся в середине или пока указатель слева не окажется справа от указателя справа. В этот момент все элементы, меньшие или равные опорному, оказываются слева от опорного элемента, а все элементы, большие опорного, - справа от него. Затем рекурсивно выполняется быстрая сортировка для обеих подгрупп.

#### **2.2.3 Метод «сортировка Шела»**

Сортировка Шелла (Shell Sort) - это алгоритм сортировки, который является усовершенствованным методом сортировки вставками. Алгоритм сортировки Шелла работает по следующей схеме:

1. Выбирается интервал (gap), с помощью которого будет осуществляться сортировка элементов в массиве. Обычно первоначальное значение интервала равно n/2, где n - количество элементов в массиве.

2. Сравниваются и переставляются элементы на расстоянии gap друг от друга.

3. Интервал gap уменьшается и сортировка повторяется для нового значения gap.

4. Сортировка продолжается, пока интервал gap не станет равен 1.

5. Выполняется сортировка вставками для отсортированного массива.

Суть алгоритма заключается в том, что сортировка Шелла пытается уменьшить количество перестановок элементов в массиве, которые необходимы для того, чтобы элементы стали отсортированными. Использование большого значения интервала позволяет переставлять элементы на большие расстояния, тем самым быстрее достигая определенной степени упорядоченности в массиве.

## **2.3 Постановка задачи**

Написать программу для исследования улучшенных методов сортировки с помощью массивов в среде Microsoft Visual Studio 2022, создать вспомогательные методы для реализации и последующей интеракции с массивами, реализовать возможность сравнения с другими методами сортировок.

Назначение: исследование улучшенных методов сортировок, сравнительный анализ, выявление эффективности.

Для разработки программы выбран объектно-ориентированный язык программирования С#. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Немаловажно, что С# является профессиональным языком, предназначенным для решения широкого спектра задач, и в первую очередь - в быстро развивающейся области создания распределенных приложений. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Выбор среды разработки обоснован тем, что Microsoft Visual Studio – одно из лучших решений для разработки приложений на языке C#, это передовая среда разработки на языке C#, предназначенная для создания интерактивных приложений с пользовательским интерфейсом для компьютеров, рабочих станций, сенсорных дисплеев, информационных терминалов и Интернета. На сегодняшний день это довольно мощная среда программирования, она обладает богатым функционалом, быстро работает, занимает мало места.

# **3 Технология разработки программы**

## **3.1 Описание алгоритма решения**

В данной работе были применены такие алгоритмы как:

1. Алгоритм сортировки пузырьком (BubbleSort);

2. Алгоритм быстрой сортировки (QuickSort);

3. Алгоритм сортировки Шела (ShellSort);

4. Алгоритм, меняющий значение переменных местами (Swap);

5. Алгоритм инициализации массива случайно-сгенерированными значениями (ArratInit);

6. Алгоритм вывода массива в консоль (ArratPrint);

Алгоритм BubbleSort реализован следующим образом:

1. Проход по массиву сравнивает каждую пару соседних элементов и меняет их местами, если они находятся в неправильном порядке.

2. Проходы по массиву повторяются, пока все элементы не будут отсортированы.

Реализация методов QuickSort и ShellSort осуществляется с помощью описанных в теоретической части алгоритмов.

Метод Swap меняет значения 2-ух переменных местами следующим образом:

1. Создается временная переменная со значением 1-ой переменной;
2. Значение первой переменной меняется на значение 2-ой;
3. Значение 2-ой меняется на значение временной.

ArrayInit реализован следующим образом:

На вход метода передаются параметры:

1. arr - массив, который будет проинициализирован случайными числами;

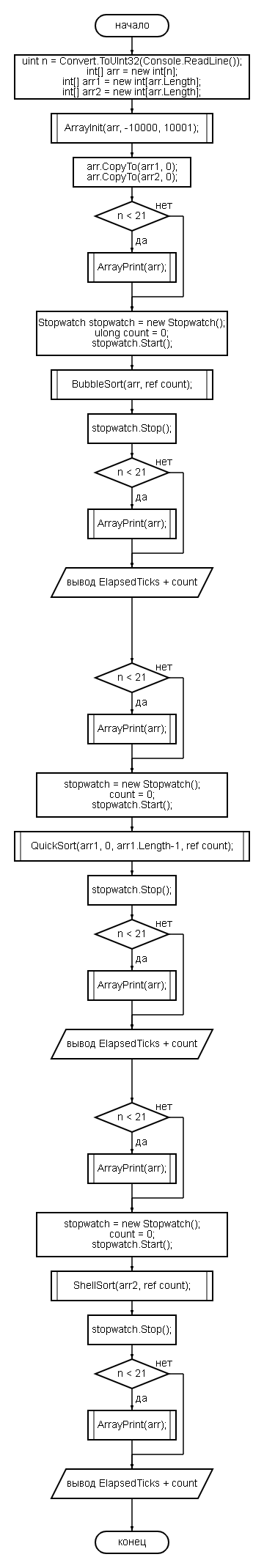
2. a - начальное значение диапазона генерируемых чисел;

3. b - конечное значение диапазона генерируемых чисел.

В методе используется класс Random для генерации случайных чисел. Цикл for проходится по всем элементам массива и для каждого элемента генерируется случайное число в диапазоне от a до b с помощью метода Next класса Random.

Метод ArrayPrint использует цикл foreach, который проходится по всем элементам массива и выводит их на экран с помощью метода Console.Write. Перед каждым элементом выводится табуляция с помощью символа \t.

## **3.2 Блок схемы алгоритма**



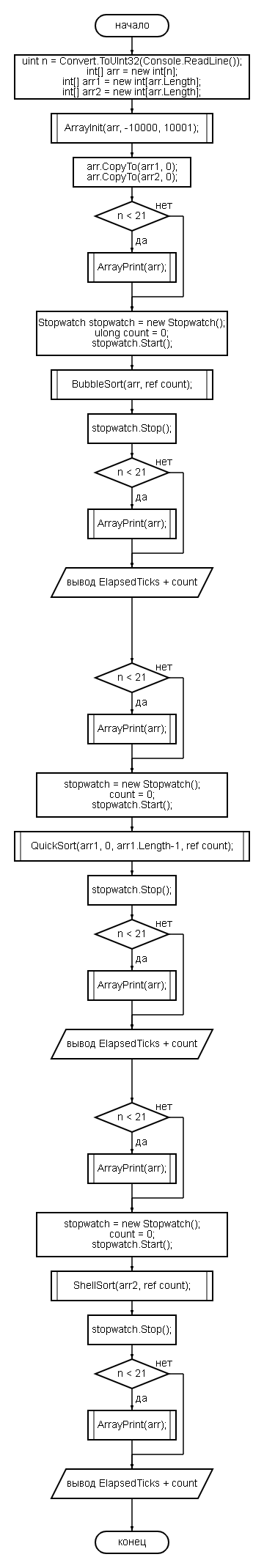


Рисунок **3.1** – Блок схема метода Main.

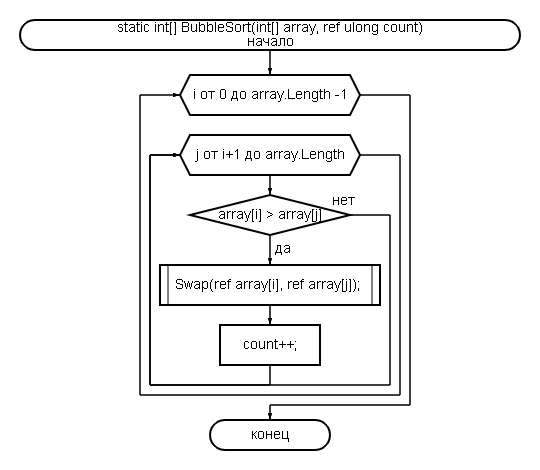


Рисунок **3.2** – Блок схема метода BubbleSort.

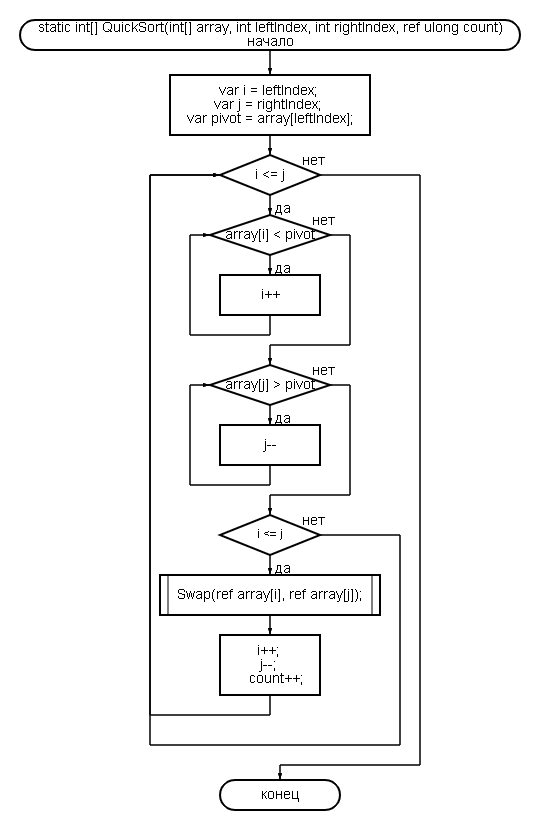


Рисунок **3.3** – Блок схема метода QuickSort.

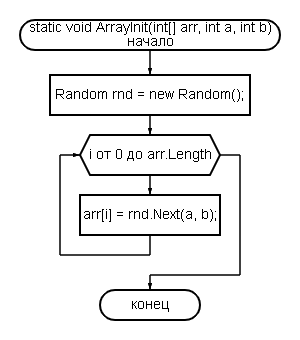


Рисунок **3.4** – Блок схема метода ArrayInit

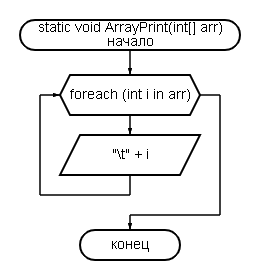


Рисунок **3.5** – Блок схема метода ArrayPrint.

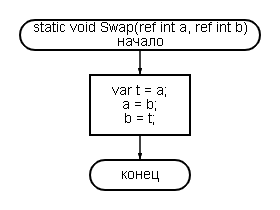


Рисунок **3.6** – Блок схема метода Swap.

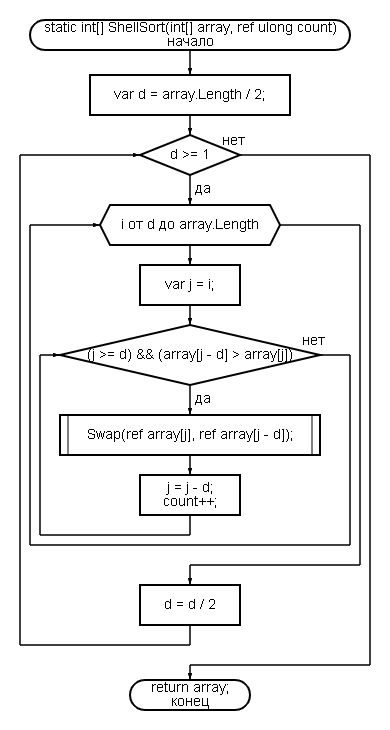


Рисунок **3.7** – Блок схема метода ShellSort.

## **3.3 Программная реализация**

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Diagnostics;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

Console.Write("\n\tВведите длину массива n: \n\t"); // если n > 0, то вывод массива не осуществляется

uint n = Convert.ToUInt32(Console.ReadLine());

int[] arr = new int[n];

int[] arr1 = new int[arr.Length];

int[] arr2 = new int[arr.Length];

ArrayInit(arr, -10000, 10001);

arr.CopyTo(arr1, 0);

arr.CopyTo(arr2, 0);

Console.WriteLine("\n\tСортировка пузырьком " + arr.Length +" элементов.");

if (n < 21)

{

Console.WriteLine("\tИзначальный массив: ");

ArrayPrint(arr);

Console.WriteLine("\n\tОтсортированный массив: ");

}

Stopwatch stopwatch = new Stopwatch();

ulong count = 0;

stopwatch.Start();

BubbleSort(arr, ref count);

stopwatch.Stop();

if (n < 21)

ArrayPrint(arr);

Console.WriteLine("\n\tВремя: " + stopwatch.ElapsedTicks + "\n\tПерестановки: " + count);

Console.WriteLine();

Console.WriteLine("\n\tБыстрая сортировка " + arr1.Length + " элементов.");

if (n < 21)

{

Console.WriteLine("\tИзначальный массив: ");

ArrayPrint(arr1);

Console.WriteLine("\n\tОтсортированный массив: ");

}

stopwatch = new Stopwatch();

count = 0;

stopwatch.Start();

QuickSort(arr1, 0, arr1.Length-1, ref count);

stopwatch.Stop();

if (n < 21)

ArrayPrint(arr1);

Console.WriteLine("\n\tВремя: " + stopwatch.ElapsedTicks + "\n\tПерестановки: " + count);

Console.WriteLine();

Console.WriteLine("\n\tСортировка Шела " + arr2.Length + " элементов.");

if (n < 21)

{

Console.WriteLine("\tИзначальный массив: ");

ArrayPrint(arr2);

Console.WriteLine("\n\tОтсортированный массив: ");

}

stopwatch = new Stopwatch();

count = 0;

stopwatch.Start();

ShellSort(arr2, ref count);

stopwatch.Stop();

if (n < 21)

ArrayPrint(arr2);

Console.WriteLine("\n\tВремя: " + stopwatch.ElapsedTicks + "\n\tПерестановки: " + count);

Console.ReadKey();

Console.WriteLine();

}

static int[] BubbleSort(int[] array, ref ulong count)

{

for (int i = 0; i < array.Length - 1; i++)

for (int j = i + 1; j < array.Length; j++)

if (array[i] > array[j])

{

Swap(ref array[i], ref array[j]);

count++;

}

return array;

}

static int[] QuickSort(int[] array, int leftIndex, int rightIndex, ref ulong count)

{

var i = leftIndex;

var j = rightIndex;

var pivot = array[leftIndex];

while (i <= j)

{

while (array[i] < pivot)

{

i++;

}

while (array[j] > pivot)

{

j--;

}

if (i <= j)

{

Swap(ref array[i], ref array[j]);

i++;

j--;

count++;

}

}

if (leftIndex < j)

QuickSort(array, leftIndex, j, ref count);

if (i < rightIndex)

QuickSort(array, i, rightIndex, ref count);

return array;

}

static void ArrayInit(int[] arr, int a, int b)

{

Random rnd = new Random();

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

arr[i] = rnd.Next(a, b);

}

static void ArrayPrint(int[] arr)

{

foreach (int i in arr)

Console.Write("\t" + i);

}

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

{

var t = a;

a = b;

b = t;

}

static int[] ShellSort(int[] array, ref ulong count)

{

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;

count++;

}

}

d = d / 2;

}

return array;

}

}

}

## **3.4 Результат выполнения программы**

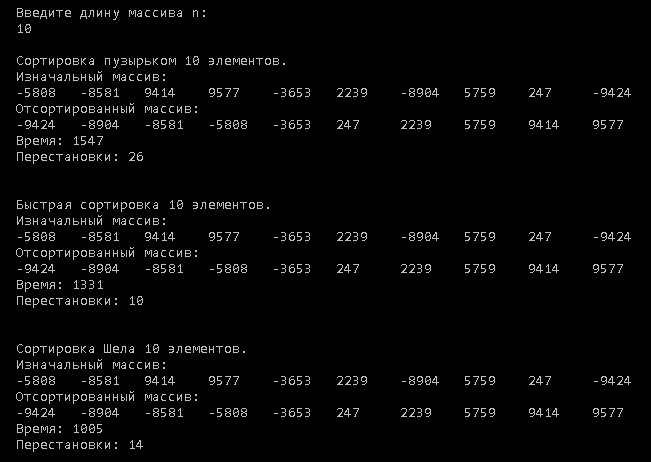


Рисунок **3.8** – Результат выполнения программы для массива длинной 10.

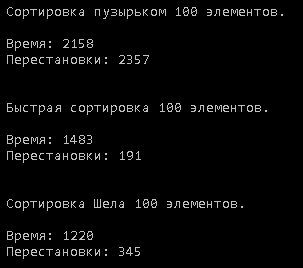


Рисунок **3.9** – Результат выполнения программы для массива длинной 100.

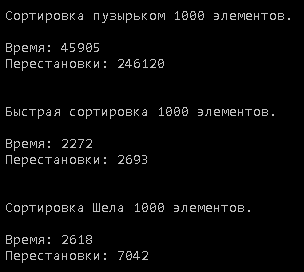


Рисунок **3.10** – Результат выполнения программы для массива длинной 1000.

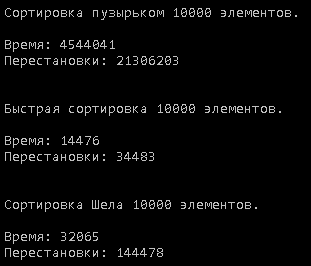


Рисунок **3.11** – Результат выполнения программы для массива длинной 10000.

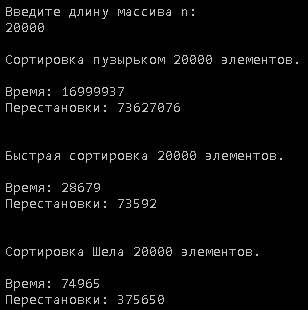


Рисунок **3.12** – Результат выполнения программы для массива длинной 20000.

## **3.5 Сравнительный анализ**

В ходе данной работы был проведен сравнительный анализ работы алгоритмов сортировки по таким критериям – затраченное время в тиках и количество перестановок.

Таблица 3.1 – «Сортировка пузырьком».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| сортировка пузырьком | количество элементов в массиве | | | | |
| 10 | 100 | 1000 | 10000 | 20000 |
| время в тиках | 1547 | 2158 | 45905 | 4544041 | 16999937 |
| перестановки | 26 | 2357 | 246120 | 21306203 | 73627076 |

Рисунок **3.13** - «Сортировка пузырьком».

Таблица 3.2 – «Быстрая сортировка».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| быстрая сортировка | количество элементов в массиве | | | | |
| 10 | 100 | 1000 | 10000 | 20000 |
| время в тиках | 1331 | 1483 | 2272 | 14476 | 28679 |
| перестановки | 10 | 191 | 2693 | 34483 | 73592 |

Рисунок. **3.14** - «Быстрая сортировка».

Таблица 3.3 – «Сортировка Шела».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| сортировка Шела | количество элементов в массиве | | | | |
| 10 | 100 | 1000 | 10000 | 20000 |
| время в тиках | 1005 | 1220 | 2618 | 32065 | 74965 |
| перестановки | 14 | 345 | 7042 | 144478 | 375650 |

Рисунок. **3.15** - «Сортировка Шела».

Таблица 3.4 – «Сравнительная таблица 10 элементов».

|  |  |  |  |
| --- | --- | --- | --- |
| размер массива 10 | Сортировка пузырьком | Быстрая сортировка | Сортировка Шела |
| время в тиках | 1547 | 1331 | 1005 |
| перестановки | 26 | 10 | 14 |

Таблица 3.5 – «Сравнительная таблица 100 элементов».

|  |  |  |  |
| --- | --- | --- | --- |
| размер массива 100 | Сортировка пузырьком | Быстрая сортировка | Сортировка Шела |
| время в тиках | 2158 | 1483 | 1220 |
| перестановки | 2357 | 191 | 345 |

Таблица 3.6 – «Сравнительная таблица 1000 элементов».

|  |  |  |  |
| --- | --- | --- | --- |
| размер массива 1000 | Сортировка пузырьком | Быстрая сортировка | Сортировка Шела |
| время в тиках | 45905 | 2272 | 2618 |
| перестановки | 246120 | 2693 | 7042 |

Таблица 3.7 – «Сравнительная таблица 10000 элементов».

|  |  |  |  |
| --- | --- | --- | --- |
| размер массива 10000 | Сортировка пузырьком | Быстрая сортировка | Сортировка Шела |
| время в тиках | 4544041 | 14476 | 32065 |
| перестановки | 21306203 | 34483 | 144478 |

Таблица 3.8 – «Сравнительная таблица 20000 элементов».

|  |  |  |  |
| --- | --- | --- | --- |
| размер массива 20000 | Сортировка пузырьком | Быстрая сортировка | Сортировка Шела |
| время в тиках | 16999937 | 28679 | 74965 |
| перестановки | 73627076 | 73592 | 375650 |

Из приведённых выше таблиц можно сделать вывод, что при увеличении объёма изначально генерируемого массива самым быстрым способом сортировки оказался метод «быстрой сортировки». Однако следует заметить, что «сортировка Шела» в среднем показывала немного лучшее результаты по времени, когда количество элементов в массиве было чуть меньше 1000 (до 100 точно), но при этом сильно проигрывало «быстрой сортировки» в перестановках. Можно предположить, что лучше использовать «сортировку Шела» вместо «быстрой сортировки», когда:

1. Основной показатель эффективности – время сортировки;

2. В сортируемом списке до 1000 элементов.

Например: упорядочить небольшой список в множестве списков.

Алгоритм «сортировки пузырьком» очень сильно проигрывал обоим алгоритмам, хотя и не сильно отставал по времени в массивах длинной до 100 элементов.

Также следует учесть, что во всех методах сортировки использовался специально-созданный метод Swap, который хоть и незначительно, но замедлял алгоритм сортировки (тоже самое касается счетчика count), поэтому результаты выполнения программы следует воспринимать как близкие к среднему показателю.

**Заключение**

Целью работы являлась разработка программы для исследования улучшенных методов сортировки с помощью одномерных массивов в среде разработки Microsoft Visual Studio 2022, цель достигнута полностью. В качестве инструмента использовалась Microsoft Visual Studio 2022. Задачи курсового проекта, реализация и анализ были успешно выполнены, в ходе работы над поставленной задачи были улучшены навыки программирования. Во время выполнения работы была написана программа, задачей которой было реализация методов улучшенной сортировки с помощью массивов. Для решения задачи были изучены методы сортировок, теоретический материал.

При выполнении курсового проекта произведено углубление в информационные источники по алгоритмам и структурам данных с целью анализа состояния решаемой задачи.

Был проведен анализ предметной области, выявлены требования к разрабатываемой программе, была разработана и реализована программа, определена эффективность разработки.

**Список использованных источников**

1. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. ‒ СПб.: Питер, 2014. ‒ 432 с.: ил.

2. Марков В.Н. Алгоритмы и структуры данных: учеб. пособие/ В.Н. Марков. ‒ Краснодар: Изд. ФГБОУ ВО «КубГТУ», 2022. ‒ 207 с.

3. Троелсен Эндрю, Джепикс Филипп Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд.: Пер. с англ. ‒ СПб.: ООО «Диалектика», 2018 ‒ 1328 с.: ил.

**Приложение А. Тест на антиплагиат**

