Реферат

Пояснительная записка к курсовой работе содержит: 34 с., 17 рис., 3 источника.

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

Цель исследования методов сортировки обменом 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 следующие алгоритма сортировки «сортировка пузырьком», «быстрая сортировка» и «сортировка расческой». В коде программы реализовано сравнение эффективности алгоритмов друг с другом.

Объект исследования: методы сортировки обменом.

Предмет исследования: «сортировка пузырьком», «быстрая сортировка», «сортировка расческой».

Содержание

Введение

Методы сортировки обменом являются одними из самых простых и популярных алгоритмов сортировки. Их основное преимущество заключается в том, что они применимы к различным типам данных, включая числа, строки, массивы и т.д. В данном контексте методы сортировки обменом используются для упорядочивания массивов чисел.

Существует множество различных методов сортировки обменом, каждый из которых имеет свои преимущества и недостатки в зависимости от особенностей данных, которые нужно отсортировать. В данном коде реализованы три метода сортировки обменом: сортировка пузырьком (Bubble Sort), быстрая сортировка (Quick Sort) и сортировка расческой (Comb Sort).

Цель данной программы - произвести исследование эффективности и сравнительный анализ трех методов сортировки обменом на основе количества обменом и времени выполнения на различных размерах массивов. В результате исследования будет получена информация о том, какой из методов сортировки обменом является наиболее эффективным в определенных условиях.

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 методов сортировки обменом в среде Microsoft Visual Studio 2022, с помощью одномерных массивов.

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

2.2 Сведения из теории

2.2.1 Методы сортировки обменом

Методы сортировки обменом — это алгоритмы сортировки элементов в массиве или списке, основанные на сравнении двух соседних элементов и их перестановке, если они находятся в неправильном порядке. Эти методы получили название «сортировки перестановками», так как они переставляют элементы массива между собой, чтобы добиться правильного порядка.

Существует несколько методов сортировки обменом, включая пузырьковую сортировку (Bubble Sort), шейкерную сортировку (Cocktail Sort), сортировку расческой (Comb Sort) и быструю сортировку (Quick Sort). Каждый из этих методов имеет свои преимущества и недостатки, а также свой оптимальный набор условий, при которых они могут работать наиболее эффективно.

Сортировки обменом часто используются в программировании, так как они просты в реализации и могут быть использованы для сортировки различных типов данных, включая числа, строки и объекты. Однако, из-за своей низкой производительности на больших объемах данных, сортировки обменом редко используются в крупномасштабных системах.

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 расческой»

Метод сортировки расческой (Comb Sort) – это усовершенствованный алгоритм сортировки пузырьком. Его особенность заключается в том, что он использует больший шаг для перемещения элементов, чем сортировка пузырьком. Это позволяет уменьшить количество итераций сортировки и увеличить скорость ее выполнения.

Метод сортировки расческой получил свое название из-за того, что он работает по принципу "расчесывания" элементов массива. При этом шаг, с

которым элементы перемещаются, постепенно уменьшается до минимального значения. Когда шаг становится равным единице, метод переключается на обычную сортировку пузырьком для окончательной сортировки.

Время выполнения метода сортировки расческой зависит от выбранного шага. Наилучшее время достигается, когда шаг равен приблизительно 1,3. Таким образом, метод сортировки расческой имеет лучшую производительность по сравнению со стандартной сортировкой пузырьком, но по-прежнему не является самым эффективным методом сортировки.

2.3 Постановка задачи

Написать программу для исследования методов сортировки обменом 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, создать удобный пользовательский интерфейс.

Назначение: исследования методов сортировки обменом, сравнительный анализ, выявление эффективности.

Для разработки программы выбран объектно-ориентированный язык программирования C#. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Немаловажно, что C# 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 расческой» (CombSort);

4. Алгоритм, генерирующий случайные значения массива (ArratInitialize);

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 CombSort осуществляется с помощью описанных в теоретической части алгоритмов.

ArratInitialize 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.

3.2 Макет программы

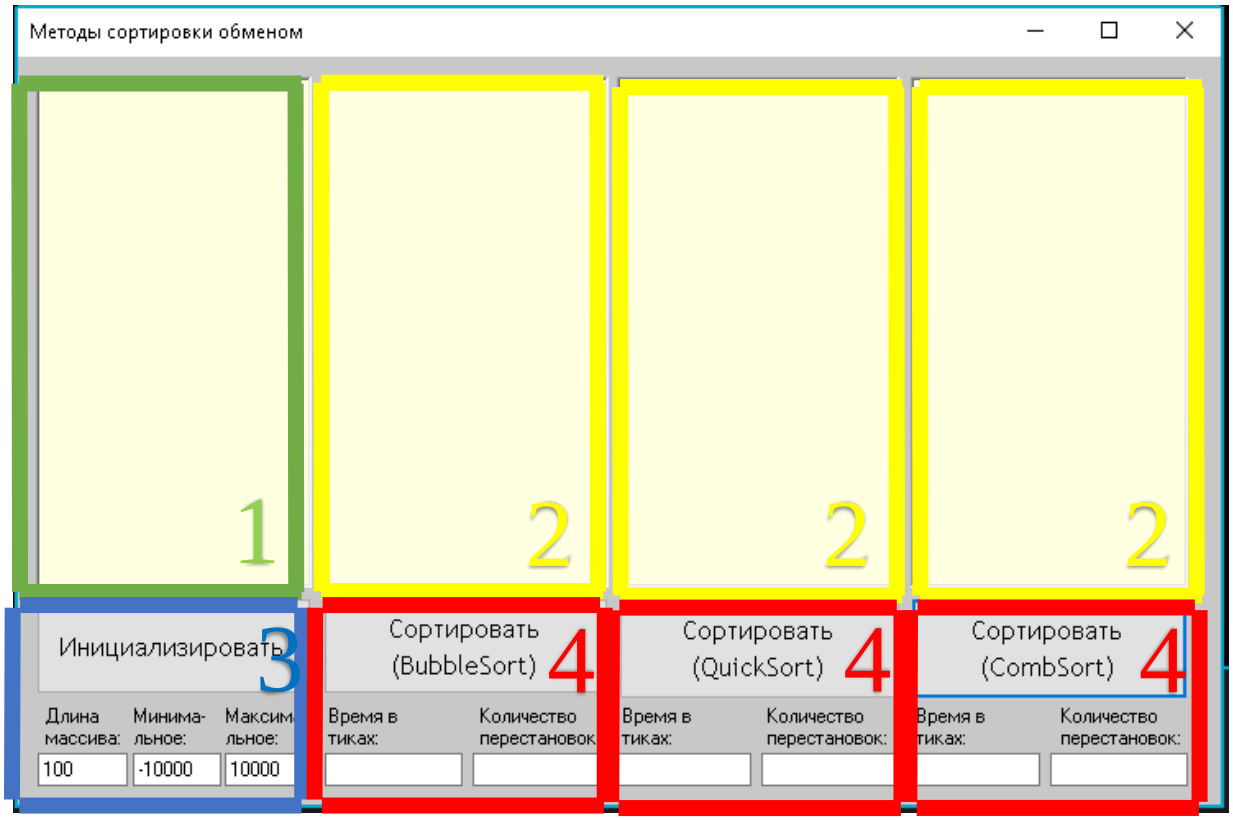

Рисунок **1** – Макет программы «Методы сортировки обменом».

На Рисунке 1 отображён макет программы «Методы сортировки обменом», где каждой цифре на рисунке соответствует свой функционал:

1. Блок вывода значений начального массива в неотсортированном виде;
2. Блок вывода значений отсортированного массива способом, указанным в кнопке под блоком;
3. Блок, включающий в себе поле ввода максимального, минимального значения и длины массива (значения по умолчанию: 100, -10000, 10000) и кнопку «Инициализировать», генерирующую массив;
4. Блок, включающий в себе кнопку «Сортировать {Метод}», где {Метод} – один из указанных алгоритмов, и поля вывода времени в тиках и количества перестановок. Кнопка сортирует изначальный массив в соответствии с указанным методом и выводит время в тиках,

количество перестановок в текстовые поля из этого блока и отсортированный массив в соответствующий блок 2.

3.3 Описание программы

Данная программа на языке C# реализует три алгоритма сортировки - пузырьковую сортировку, быструю сортировку и сортировку расческой. Она имеет графический интерфейс пользователя (GUI), созданный с помощью Windows Forms, и содержит различные поля ввода.

При нажатии на кнопку "Initialize" создается массив целых чисел заданной длины, элементы которого инициализируются случайными числами в заданном диапазоне. Массив отображается в элементе управления "richTextBox1".

При нажатии на кнопку "BubbleSort" выполняется пузырьковая сортировка созданного массива. Отсортированный массив отображается в элементе управления "richTextBox2", а время выполнения сортировки и количество перестановок элементов выводятся в соответствующих полях.

При нажатии на кнопку "QuickSort" выполняется быстрая сортировка созданного массива. Отсортированный массив отображается в элементе управления "richTextBox3", а время выполнения сортировки и количество перестановок элементов выводятся в соответствующих полях.

При нажатии на кнопку "CombSort" выполняется сортировка расческой созданного массива. Отсортированный массив отображается в элементе управления "richTextBox4", а время выполнения сортировки и количество перестановок элементов выводятся в соответствующих полях.

V

1.4 Результат выполнения программы

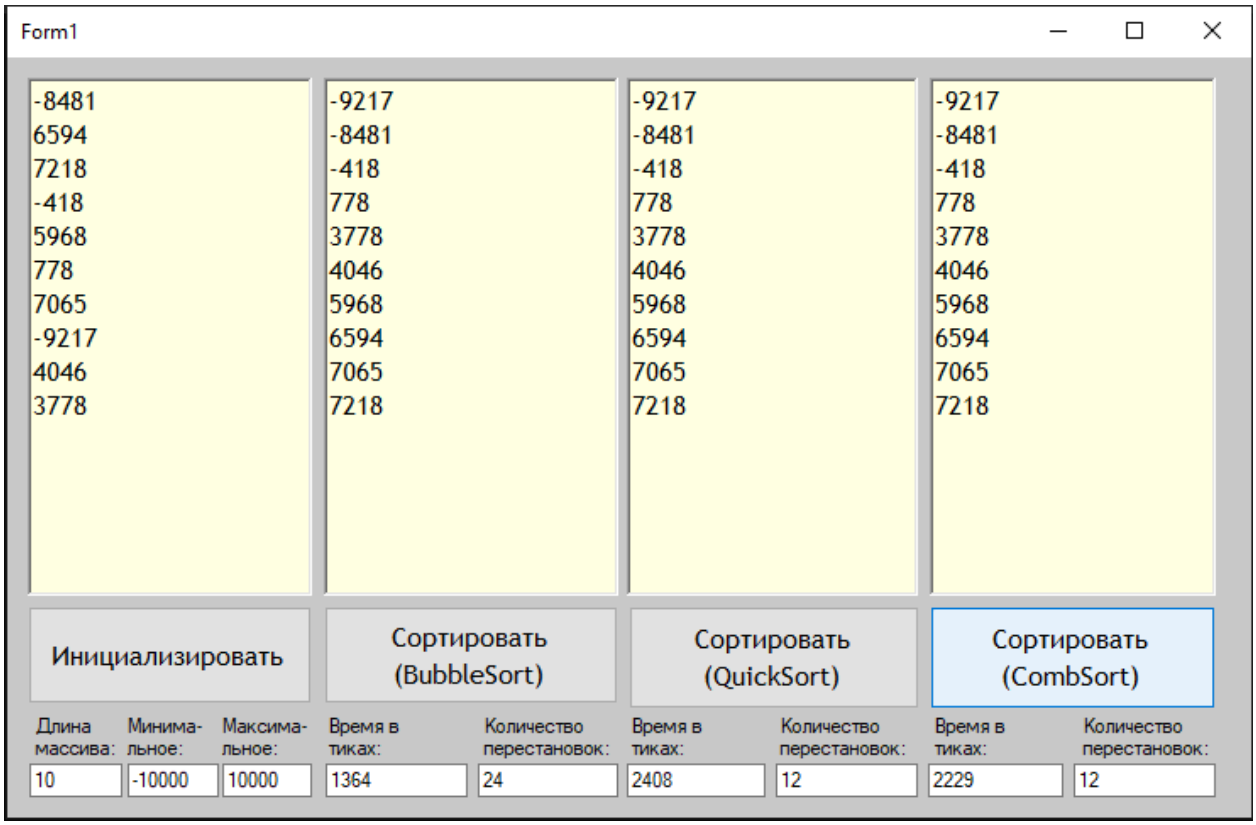

Рисунок **2** – Результат выполнения программы для массива длинной 10.

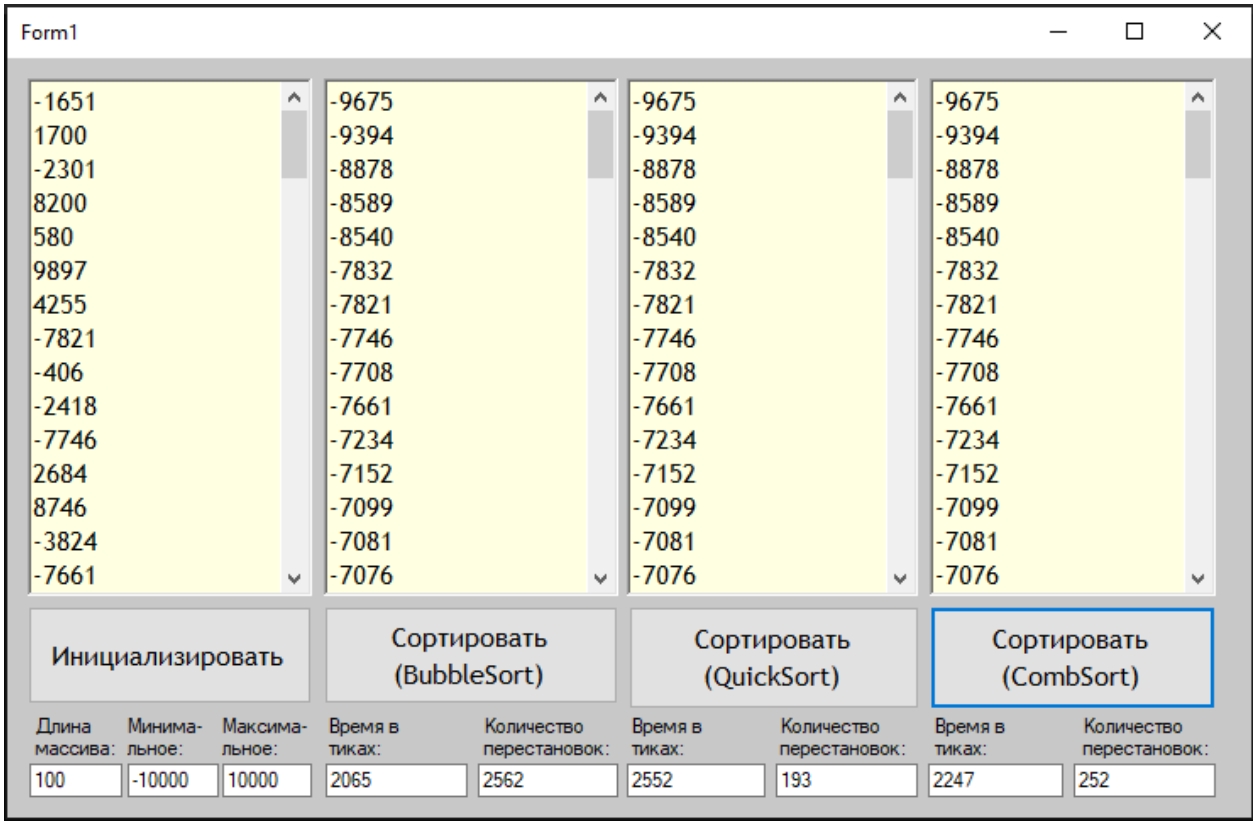

Рисунок **3** – Результат выполнения программы для массива длинной 100.

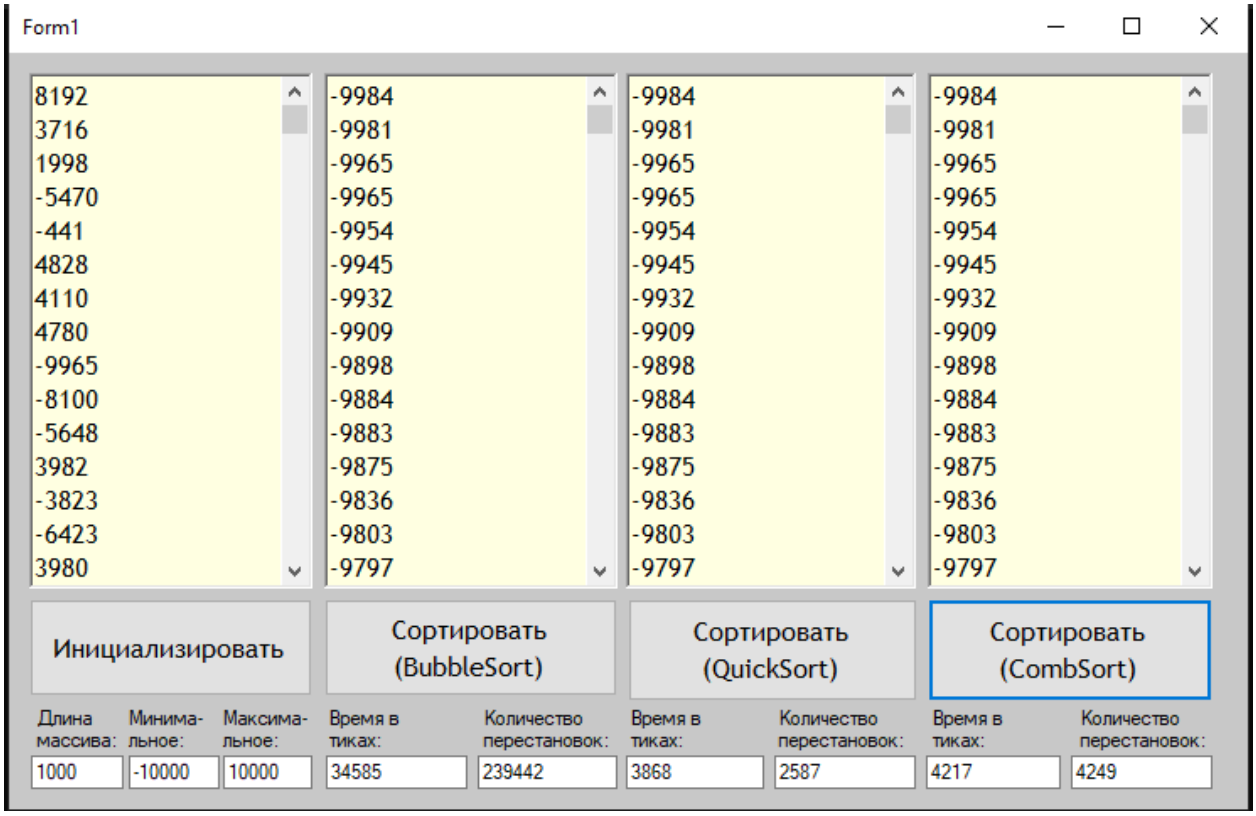

Рисунок **4** – Результат выполнения программы для массива длинной 1000.

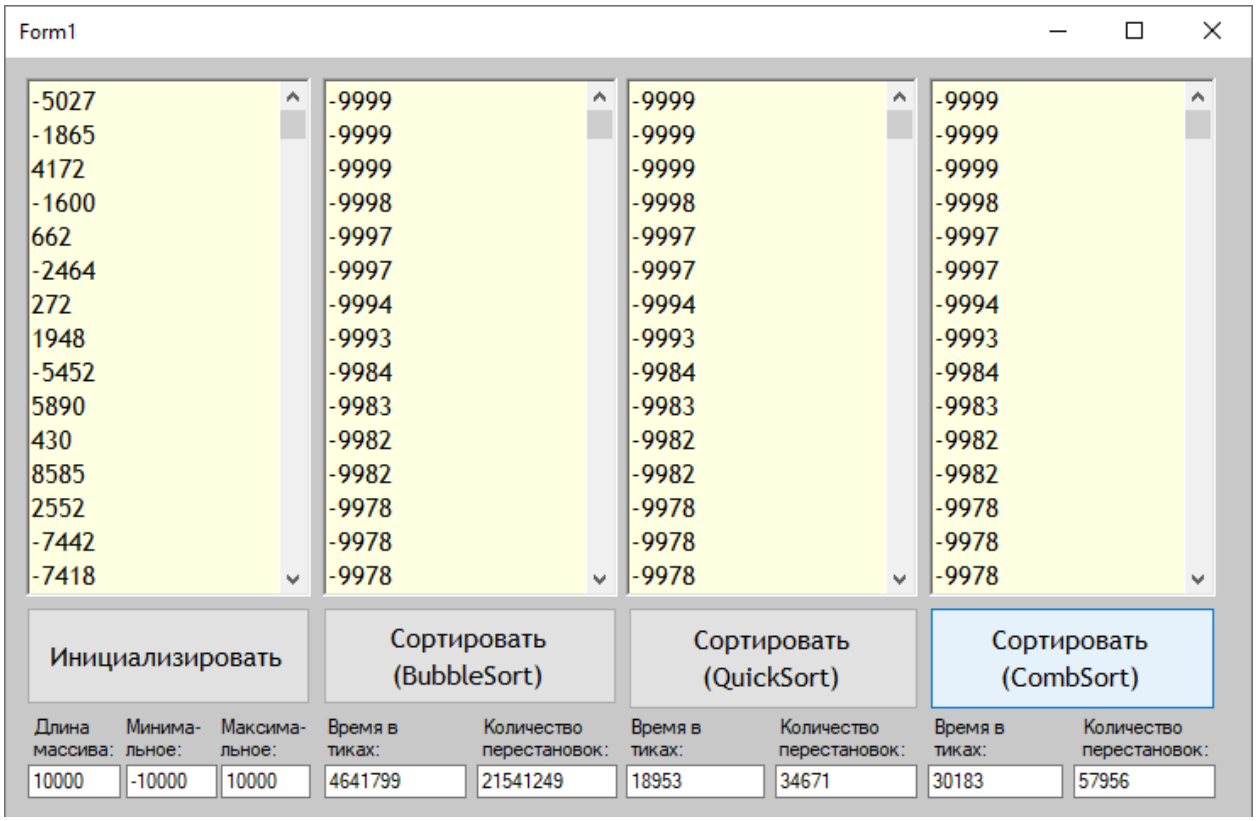

Рисунок **5** – Результат выполнения программы для массива длинной 10000.

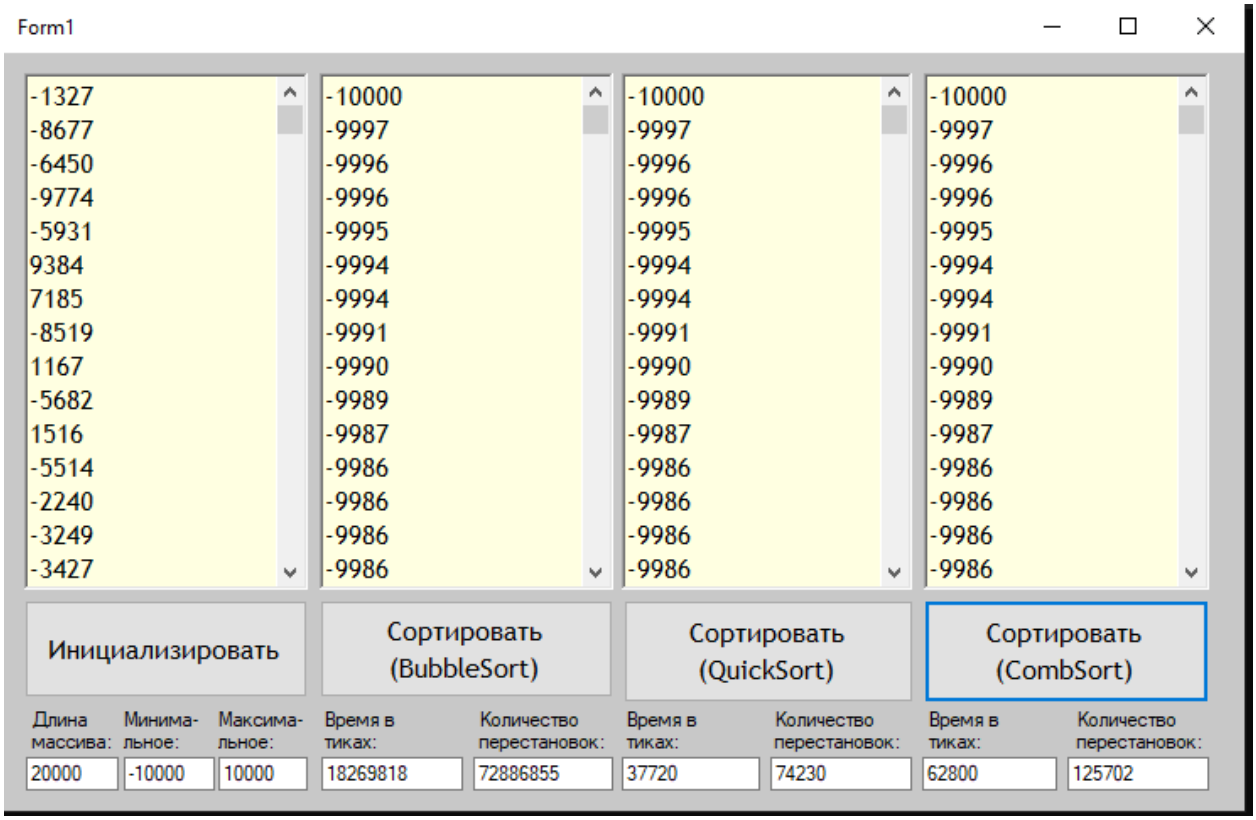

Рисунок **6** – Результат выполнения программы для массива длинной 20000.

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 1 – «Сортировка пузырьком».

сортировка пузырьком	количество элементов в массиве				
	10	100	1000	10000	20000
время в тиках	1364	2065	34585	4641799	18269818
перестановки	24	2562	239442	21541249	72886855

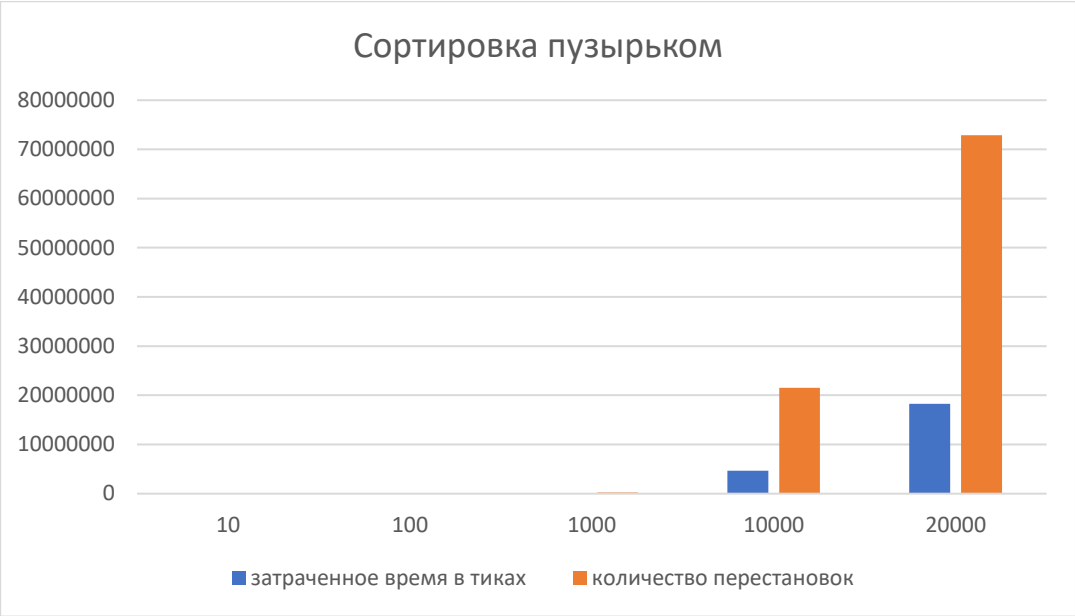

Рисунок **7** - «Сортировка пузырьком».

Таблица 2 – «Быстрая сортировка».

быстрая сортировка	количество элементов в массиве				
	10	100	1000	10000	20000
время в тиках	2408	2552	3868	18953	37720
перестановки	12	193	2587	34671	74230

Рисунок **8** - «Быстрая сортировка».

Таблица 3 – «Сортировка расческой».

сортировка расческой	количество элементов в массиве				
	10	100	1000	10000	20000
время в тиках	2229	2247	4217	30183	62800
перестановки	12	252	4249	57956	125702

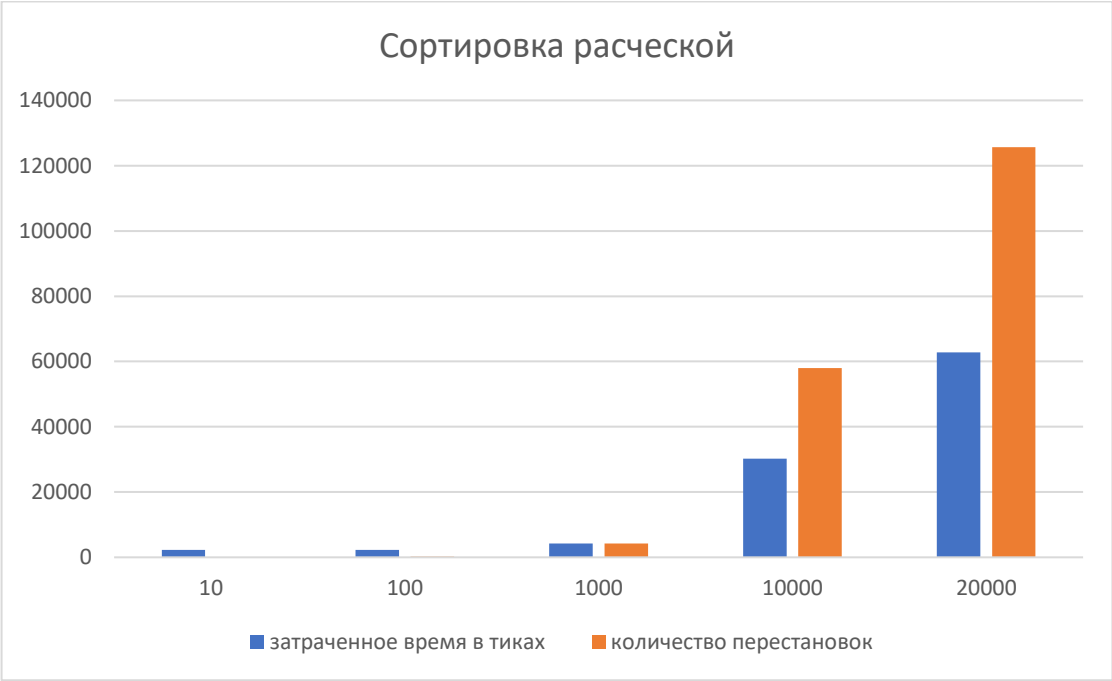

Рисунок **9** - «Сортировка расческой».

Таблица 4 – «Сравнительная таблица 10 элементов».

размер массива 10	Сортировка пузырьком	Быстрая сортировка	Сортировка расческой
время в тиках	1364	2408	2229
перестановки	24	12	12

Таблица 5 – «Сравнительная таблица 100 элементов».

размер массива 100	Сортировка пузырьком	Быстрая сортировка	Сортировка расческой
время в тиках	2065	2552	2247
перестановки	2562	193	252

Таблица 6 – «Сравнительная таблица 1000 элементов».

размер массива 1000	Сортировка пузырьком	Быстрая сортировка	Сортировка расческой
время в тиках	34585	3868	4217
перестановки	239442	2587	4249

Таблица 7 – «Сравнительная таблица 10000 элементов».

размер массива 10000	Сортировка пузырьком	Быстрая сортировка	Сортировка расческой
время в тиках	4641799	18953	30183
перестановки	21541249	34671	57956

Таблица 8 – «Сравнительная таблица 20000 элементов».

размер массива 20000	Сортировка пузырьком	Быстрая сортировка	Сортировка расческой
время в тиках	18269818	37720	62800
перестановки	72886855	74230	125702

Из приведенных выше сравнительны таблиц, можно сделать вывод, что что при увеличении объёма изначально генерируемого массива наибольшей эффективностью обладает метод «быстрой сортировки». В среднем результаты не отставали от других методов, а на массивах с большим количеством элементов разница между эффективностью методов была колоссальной.

Метод «сортировки расческой» показывал чуть лучшее результаты по времени на массивах, величина которых была меньше 1000, но при этом отставая по количеству перестановок.

«Сортировка пузырьком» оказалась еще быстрее, но проявлялось это на массивах размером меньше 100, и при этом количество перестановок было намного больше, чем в других методах.

Можно предположить, что наилучшей ситуацией для использования «сортировки пузырьком» или «сортировки расческой» вместо «быстрой сортировки» является следующей:

1. Главный критерий – скорость сортировки, при этом количество перестановок не учитывается.
2. Размер массива меньше 100 элементов.

Также следует учесть, что во всех методах использовался счетчик «Counter», который хоть и незначительно, но увеличивал показатель времени в тиках, поэтому результаты выполнения программы следует воспринимать как близкие к среднему показателю.

Заключение

Целью работы являлась разработка программы для исследования методов сортировки обменом 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, было спроектировано и реализовано приложение, 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

Приложение 1: Тест антиплагиат.

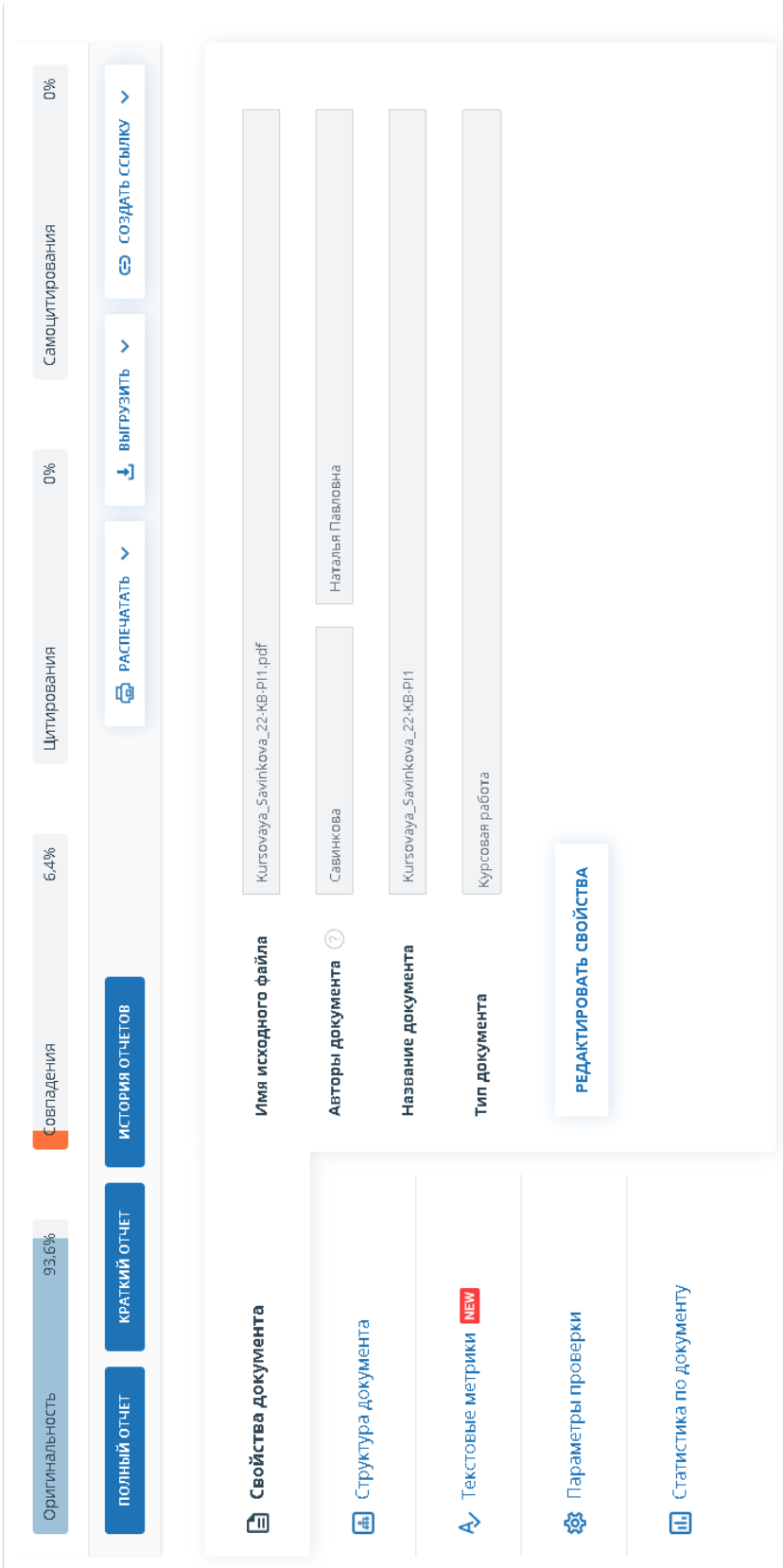

Рисунок **10** – Тест Антиплагиат.

Приложение 2: Код программы.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace ConsoleApp1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      int[] array;
      private void buttonInitialize_Click(object sender, EventArgs e)
      {
         array = new int[Convert.ToInt32(textBoxLength.Text)];
         ArrayInitialize(array, Convert.ToInt32(textBoxMin.Text),
Convert.ToInt32(textBoxMax.Text));
         string str = "";
         foreach (int i in array)
            str += i + "\n";
```

```
    richTextBox1.Text = str;
}
private void buttonBubbleSort_Click(object sender, EventArgs e)
{
    int[] array_new1 = new int[array.Length];
    Array.Copy(array, array_new1, array.Length);
    ulong Counter = 0;
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    BubbleSort(array_new1, ref Counter);
    stopwatch.Stop();
    string str = "";
    foreach (int i in array_new1)
        str += i + "\n";
    richTextBox2.Text = str;
    timeBubbleSort.Text = stopwatch.ElapsedTicks.ToString();
    permutationBubbleSort.Text = Counter.ToString();
}
private void buttonQuickSort_Click(object sender, EventArgs e)
{
    int[] array_new2 = new int[array.Length];
    Array.Copy(array, array_new2, array.Length);
    ulong Counter = 0;
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    QuickSort(array_new2, 0, array_new2.Length - 1, ref Counter);
    stopwatch.Stop();
    string str = "";
    foreach (int i in array_new2)
        str += i + "\n";
```

```
    richTextBox3.Text = str;
    timeQuickSort.Text = stopwatch.ElapsedTicks.ToString();
    permutationQuickSort.Text = Counter.ToString();
}
private void buttonCombSort_Click(object sender, EventArgs e)
{
    int[] array_new3 = new int[array.Length];
    Array.Copy(array, array_new3, array.Length);
    ulong Counter = 0;
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    CombSort(array_new3, ref Counter);
    stopwatch.Stop();
    string str = "";
    foreach (int i in array_new3)
        str += i + "\n";
    richTextBox4.Text = str;
    timeCombSort.Text = stopwatch.ElapsedTicks.ToString();
    permutationCombSort.Text = Counter.ToString();
}
static void ArrayInitialize(int[] arr, int min, int max)
{
    Random rnd = new Random();
    for (int i = 0; i < arr.Length; i++)
        arr[i] = rnd.Next(min, max + 1);
}
static int[] BubbleSort(int[] array, ref ulong Counter)
{
    for (int i = 0; i < array.Length - 1; i++)
        for (int j = i + 1; j < array.Length; j++)
```

```
            if (array[i] > array[j])
            {
                var t = array[i];
                array[i] = array[j];
                array[j] = t;
                Counter++;
            }
        return array;
    }
    static int[] QuickSort(int[] array, int leftIndex, int rightIndex, ref ulong
Counter)
    {
        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)
            {
                var t = array[i];
                array[i] = array[j];
```

```
            array[j] = t;
            i++;
            j--;
            Counter++;
        }
    }
    if (leftIndex < j)
        QuickSort(array, leftIndex, j, ref Counter);
    if (i < rightIndex)
        QuickSort(array, i, rightIndex, ref Counter);
    return array;
}
public static void CombSort(int[] input, ref ulong Counter)
{
    double gap = input.Length;
    bool swaps = true;
    while (gap > 1 || swaps)
    {
        gap /= 1.247330950103979;
        if (gap < 1) { gap = 1; }
        int i = 0;
        swaps = false;
        while (i + gap < input.Length)
        {
            int igap = i + (int)gap;
            if (input[i] > input[igap])
            {
                int swap = input[i];
                input[i] = input[igap];
                input[igap] = swap;
```

```
                    swaps = true;
                    Counter++;
                }
                i++;
            }
        }


    }
  }
}
```

Приложение 3: Блок схемы.

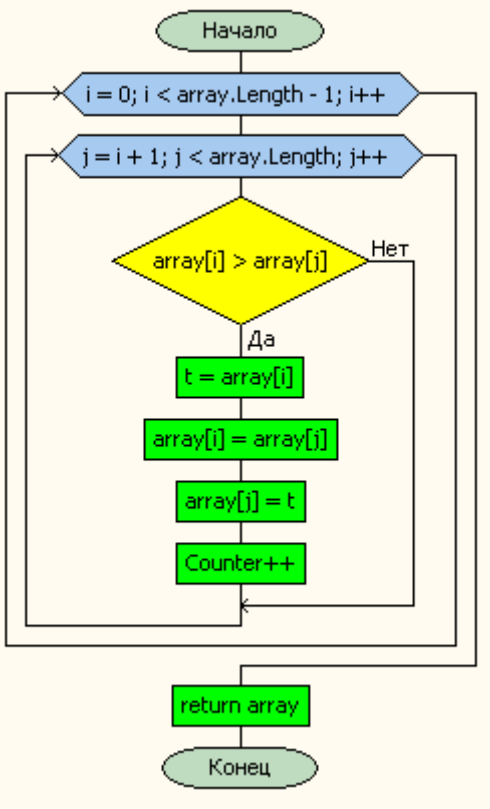

Рисунок **11** – Блок схема метода «Сортировка пузырьком».

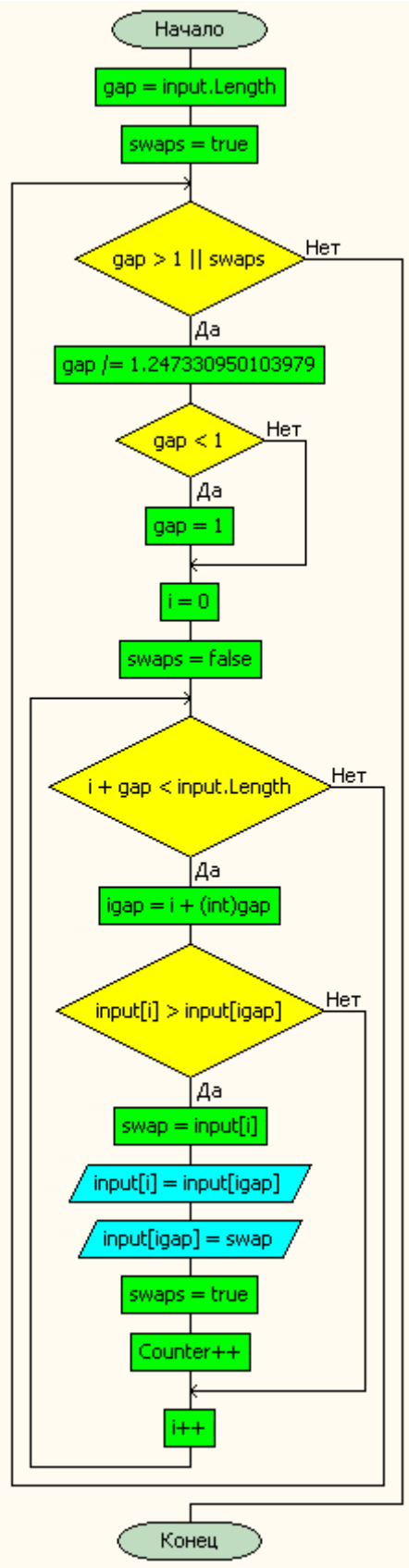

Рисунок **12** – Блок схема метода «Сортировка расческой».

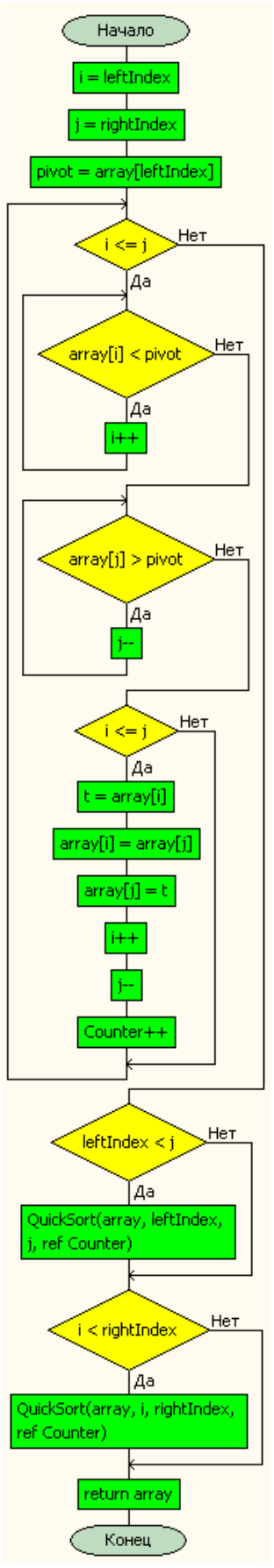

Рисунок **13** – Блок схема метода «Быстрой сортировки».

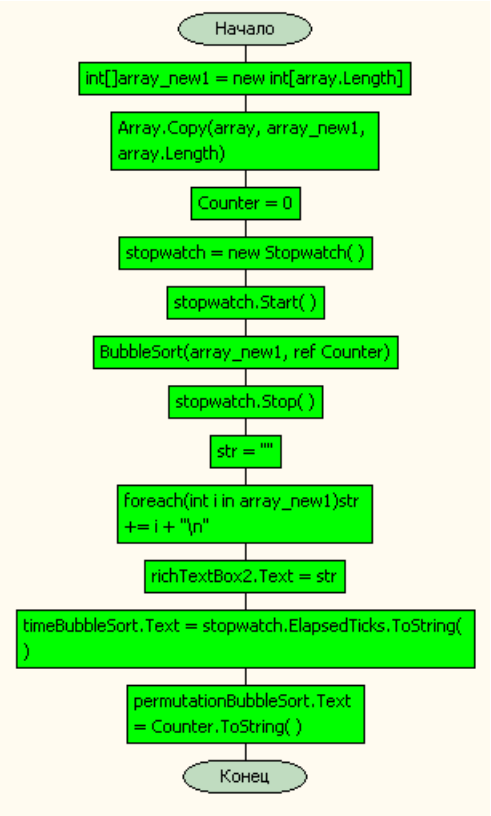

Рисунок **14** – Блок схема метода buttonBubbleSort_Click.

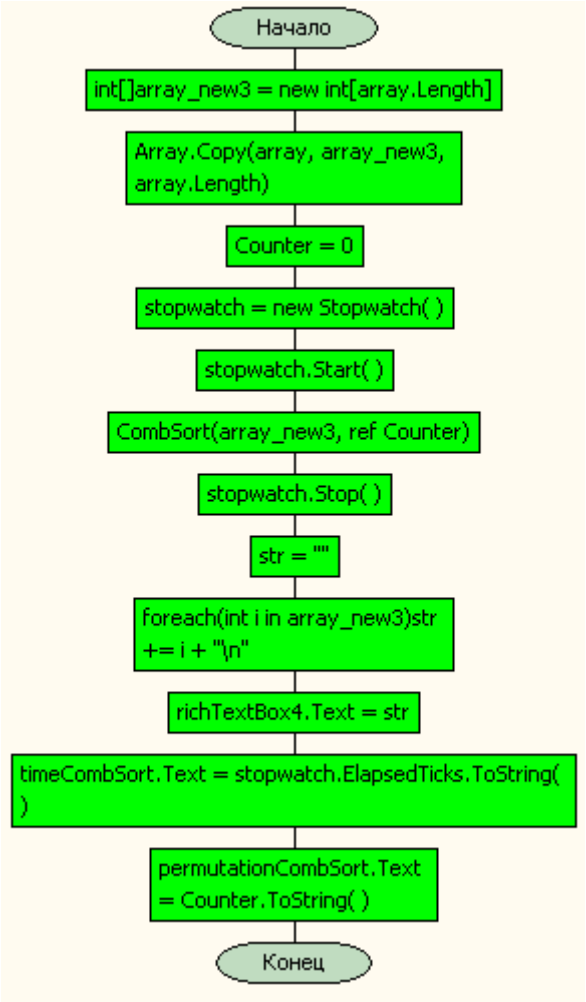

Рисунок **15** – Блок схема метода buttonCombSort_Click.

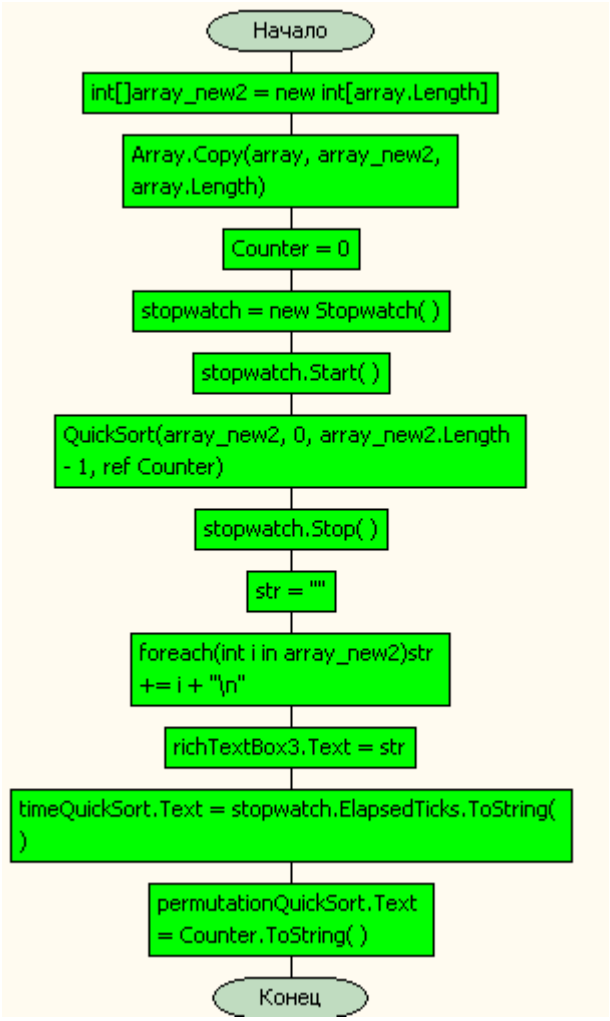

Рисунок **16** – Блок схема метода buttonQuickSort_Click.

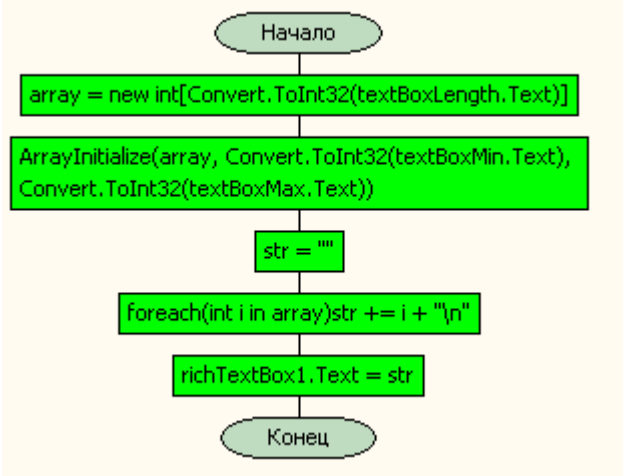

Рисунок **17** – Блок схема метода buttonInitialize_Click.