МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №10

по дисциплине

Основы алгоритмизации и программирования

Тема: «Внутренняя сортировка данных»

Работу выполнил

Студент гр.4238

Бусов В.Р.

Принял

Преподаватель Шмидт И.Р.

Казань 2024

**ВАРИАНТ 4**

**Цель работы**

Изучение простейших и улучшенных методов сортировки и особенностей их программной реализации.

**Задание на лабораторную работу**

Реализовать программу, объединяющую улучшенные и простейшие методы сортировки массивов.

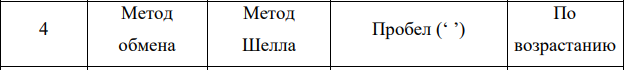
Систему взаимодействия алгоритмов сортировки и сортируемого набора данных представить в виде паттерна «Стратегия».

Каждый метод сортировки инкапсулируется в свой класс, добавляемый в основной проект по мере разработки. Также необходим вспомогательный метод генерации исходного массива случайных целых чисел с заданным числом элементов и выводом этого массива на экран.

Создать форму для демонстрации работы алгоритма сортировки. На каждом шаге отметить сравниваемые элементы. Исходный массив должен обрабатываться методом сортировки, выбранным пользователем, с подсчетом и выводом фактического числа выполненных сравнений и пересылок.

Выполнить сортировку нескольких массивов с разным числом элементов и провести сравнительный анализ эффективности методов, который будет производиться программой с выводом формы с результатами сортировки, где будет указываться количество сравнений, перестановок и время сортировки по каждому методу с определенным количеством элементов.

Предусмотреть возможность файлового ввода-вывода неотсортированного и сортированного массива. При выводе отсортированных данных выводить в файл информацию о производимых сравнениях и перестановках.



**Результат выполнения работы**

Для сортировки методом обмена сначала сгенерируем 10 чисел, а затем нажмем на кнопку начать сортировку и получим результат (Рисунок 1).

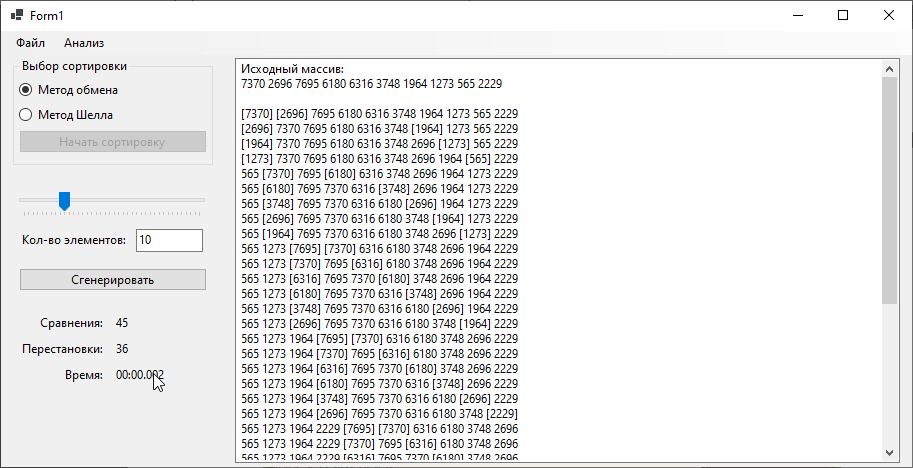


Рисунок 1 – Сортировка методом обмена

Для сортировки методом Шелла повторим те же действия и получим отсортированный массив чисел (Рисунок 2).

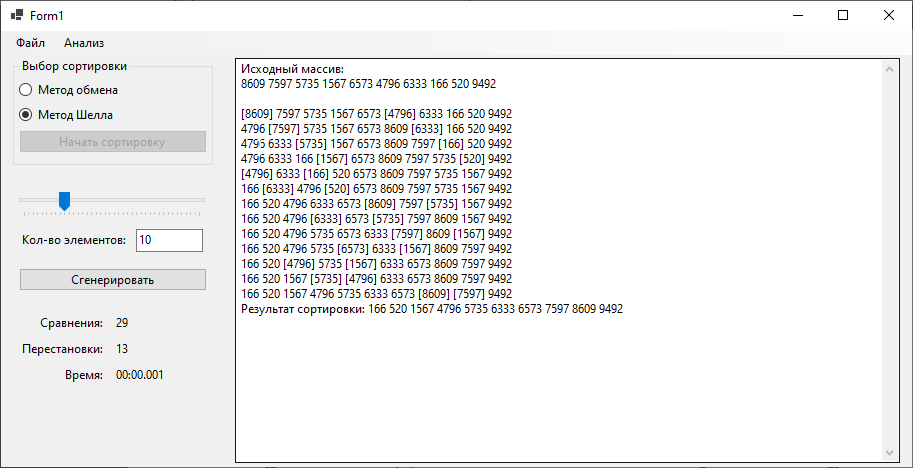


Рисунок 2 – Сортировка методом Шелла

Далее посмотрим сравнительный анализ двух методов сортировки: название, количество, сравнения, перестановки и время (Рисунок 3).

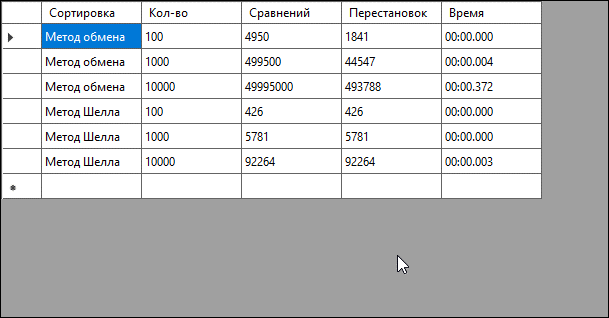


Рисунок 3 – Анализ

Далее создадим в блокноте файл, который откроем для дальнейшей сортировки (Рисунок 4).

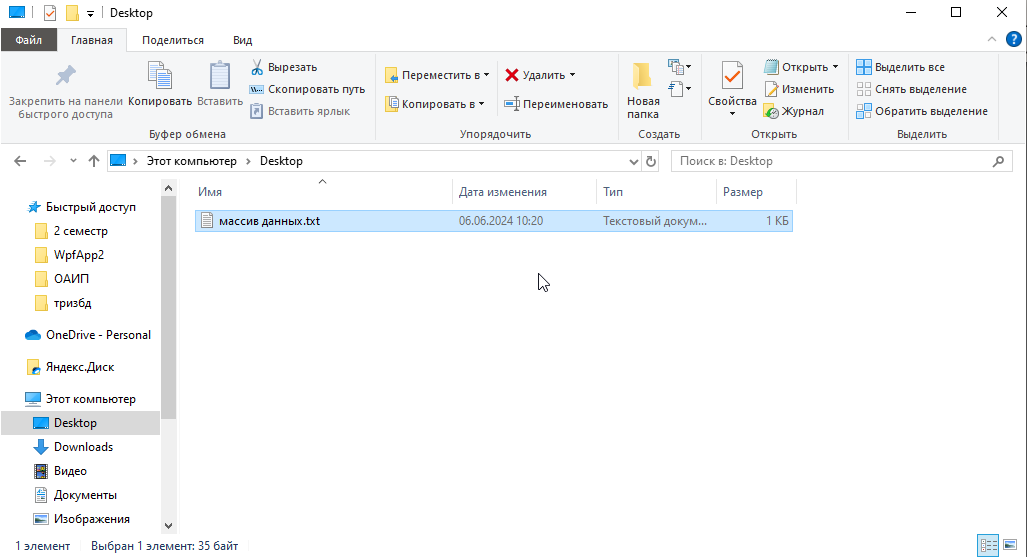


Рисунок 4 – Файл для сортировки

Далее откроем файл с числами, заранее созданный для выполнения выбранной сортировки. Выполним сортировку методом Шелла и увидим, что все отсортировано успешно (Рисунок 5).

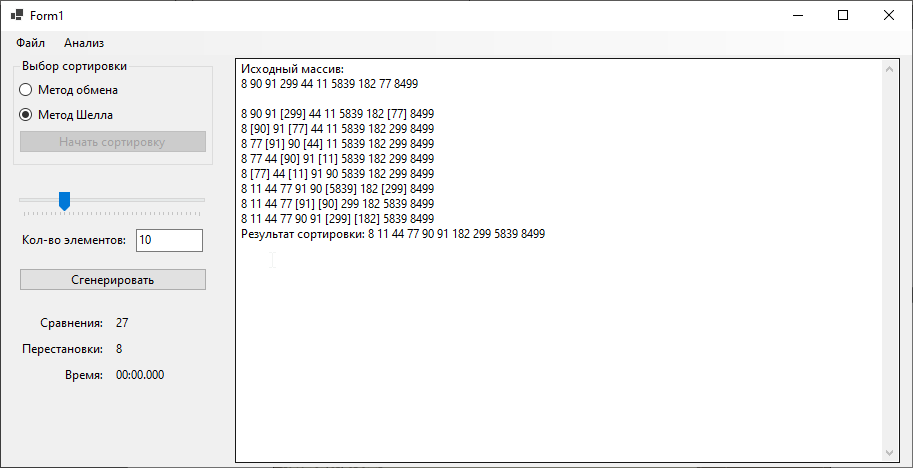


Рисунок 5 – Сортировка открытого файла

Далее попробуем снова сгенерировать несколько случайных чисел, выполнить сортировку и сохранить отсортированный файл. Затем откроем его и сверимся, что результаты совпадают (Рисунок 6).

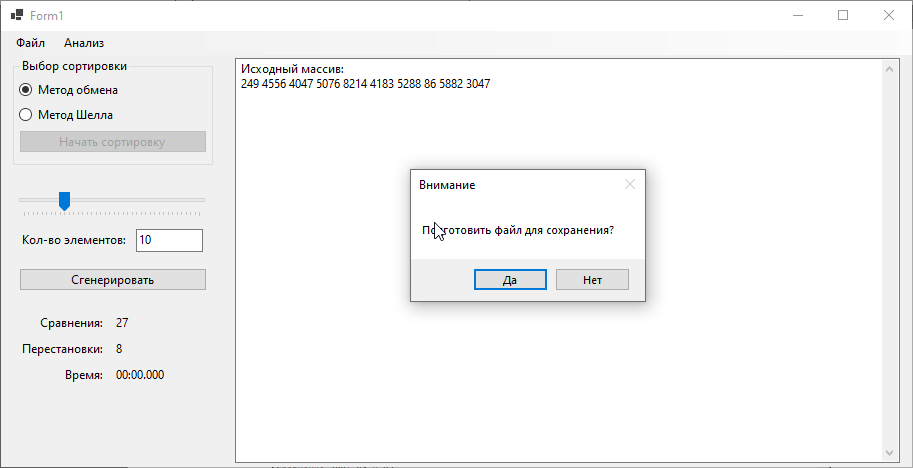


Рисунок 6 – Сохранение результата сортировки в файл

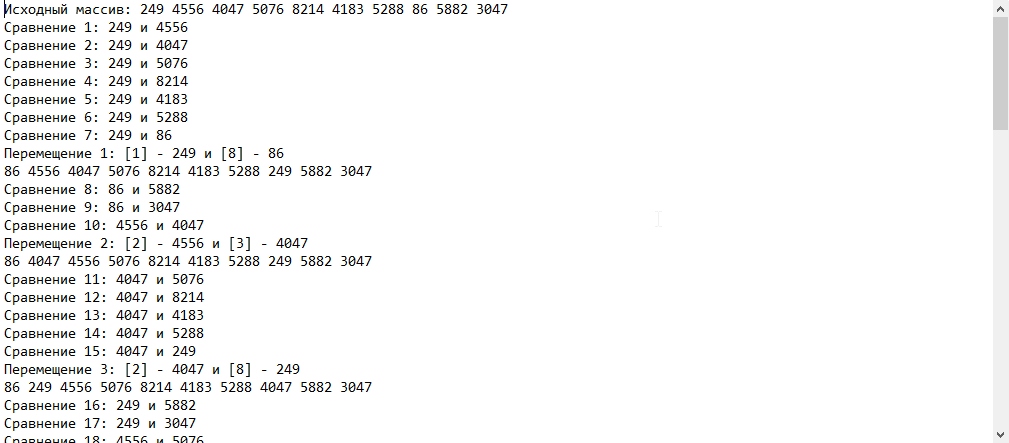


Рисунок 7 – Сохраненный в файл результат сортировки

**Листинг**

**ImprovedSort.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Drawing.Imaging;

using System.Linq;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace WinFormsApp1

{

internal class ImprovedSort: IStrategy

{

private static int comparisons = 0;

private static int permutations = 0;

private static string time = "MM:SS.MS";

private static int range = 10;

private static int length = 2;

public void SortArr(int[] mas, bool flag)

{

System.Diagnostics.Stopwatch myStopwatch = new System.Diagnostics.Stopwatch();

myStopwatch.Start();

//расстояние между элементами, которые сравниваются

var d = mas.Length / 2;

while (d >= 1)

{

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

{

var j = i;

while ((j >= d) && (Comparisons(mas, j - d, j, flag)))

{

Swap(mas, j, j - d, flag);

j = j - d;

}

}

d = d / 2;

}

myStopwatch.Stop();

var resultTime = myStopwatch.Elapsed;

time = string.Format("{0:00}:{1:00}.{2:000}", resultTime.Minutes, resultTime.Seconds, resultTime.Milliseconds);

if (flag == false)

{

FileOut.fileString = null;

}

Form1.ReadInfo(comparisons, permutations, time);

comparisons = 0;

permutations = 0;

}

private bool Comparisons(int[] arr, int ind1, int ind2, bool flag)

{

comparisons++;

if (flag && FileOut.fileString == null)

{

FileOut.fileString = "Исходный массив: ";

FileOut.Fill();

}

if (flag)

FileOut.fileString += $"Сравнение {comparisons}: " + $"{arr[ind1]} и {arr[ind2]}\n";

return arr[ind1] > arr[ind2];

}

private void Swap(int[] arr, int ind1, int ind2, bool flag)

{

permutations++;

if (flag)

FileOut.fileString += $"Перемещение {permutations}: " + $"[{ind1 + 1}] - {arr[ind1]} и [{ind2 + 1}] - {arr[ind2]}\n";

string swapString = "";

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

if (i == ind1 || i == ind2)

swapString += $"[{arr[i]}] ";

else

swapString += arr[i] + " ";

Form1.AddSortLine(swapString);

int temp = arr[ind1];

arr[ind1] = arr[ind2];

arr[ind2] = temp;

FileOut.Fill();

}

public void AnalSortArr(int[] arr, AnalInfo sort)

{

System.Diagnostics.Stopwatch myStopwatch = new System.Diagnostics.Stopwatch();

myStopwatch.Start();

//расстояние между элементами, которые сравниваются

var d = arr.Length / 2;

while (d >= 1)

{

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

{

var j = i;

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

{

int temp = arr[j];

arr[j] = arr[j - d];

arr[j - d] = temp;

comparisons++;

permutations++;

j = j - d;

}

}

d = d / 2;

}

myStopwatch.Stop();

var resultTime = myStopwatch.Elapsed;

sort.time = string.Format("{0:00}:{1:00}.{2:000}", resultTime.Minutes, resultTime.Seconds, resultTime.Milliseconds);

sort.comparisons = comparisons;

sort.permutations = permutations;

comparisons = 0;

permutations = 0;

Form1.line = "";

}

}

}

**Form2.cs**

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;

namespace WinFormsApp1

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

dataGridView1.Dock = DockStyle.Fill;

var columnType = new DataGridViewTextBoxColumn();

columnType.HeaderText = "Сортировка";

dataGridView1.Columns.Add(columnType);

var columnCount = new DataGridViewTextBoxColumn();

columnCount.HeaderText = "Кол-во";

dataGridView1.Columns.Add(columnCount);

var columnComparisons = new DataGridViewTextBoxColumn();

columnComparisons.HeaderText = "Сравнений";

dataGridView1.Columns.Add(columnComparisons);

var columnPermutations = new DataGridViewTextBoxColumn();

columnPermutations.HeaderText = "Перестановок";

dataGridView1.Columns.Add(columnPermutations);

var columnTime = new DataGridViewTextBoxColumn();

columnTime.HeaderText = "Время";

dataGridView1.Columns.Add(columnTime);

AnalInfo[] simpleSorts =

{

new AnalInfo() { count = 100 },

new AnalInfo() { count = 1000 },

new AnalInfo() { count = 10000 }

};

foreach (var sort in simpleSorts)

{

Random random = new Random();

int[] newArr = new int[sort.count];

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

newArr[i] = random.Next(0, 100);

Context.array = newArr;

new Context(new SimpleSort()).SortArr(sort);

dataGridView1.Rows.Add("Метод обмена", sort.count, sort.comparisons, sort.permutations, sort.time);

}

AnalInfo[] improvedSorts =

{

new AnalInfo() { count = 100 },

new AnalInfo() { count= 1000 },

new AnalInfo() { count = 10000 }

};

foreach (var sort in improvedSorts)

{

Random random = new Random();

int[] newArr = new int[sort.count];

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

newArr[i] = random.Next(0, 100);

Context.array = newArr;

new Context(new ImprovedSort()).SortArr(sort);

dataGridView1.Rows.Add("Метод Шелла", sort.count, sort.comparisons, sort.permutations, sort.time);

}

FileOut.fileString = null;

}

}

}

**Form1.cs**

using System.Threading;

namespace WinFormsApp1

{

public partial class Form1 : Form

{

private static int comparisons = 0;

private static int permutations = 0;

private static string time = "MM:SS:MS";

public static string line = "";

Random random = new Random();

public Form1()

{

InitializeComponent();

}

public static void AddSortLine(string str)

{

line += str + "\r\n";

}

public void AddLine()

{

textBox2.Text += line;

line = "";

}

public static void ReadInfo(int comp, int perm, string ti)

{

comparisons = comp;

permutations = perm;

time = ti;

}

public void FillLine(bool source = true)

{

if (source) this.textBox2.Text = "Èñõîäíûé ìàññèâ:\r\n";

else this.textBox2.Text += "Ðåçóëüòàò ñîðòèðîâêè: ";

foreach (var i in Context.array)

this.textBox2.Text += i + " ";

this.textBox2.Text += "\r\n\r\n";

}

private void PrintInfo()

{

comparisionLabel.Text = comparisons.ToString();

swapLabel.Text = permutations.ToString();

timeLabel.Text = time.ToString();

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

textBox1.Text = trackBar1.Value.ToString();

}

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

{

textBox2.Text = "";

button1.Enabled = true;

var newArr = new int[trackBar1.Value];

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

newArr[i] = random.Next(0, 10000);

Context.array = newArr;

FillLine();

}

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

{

if (this.textBox2.Text != "" && Context.array != null)

{

button1.Enabled = false;

FileOut.fileString = null;

var mboxResult = MessageBox.Show("Ïîäãîòîâèòü ôàéë äëÿ ñîõðàíåíèÿ?", "Âíèìàíèå", MessageBoxButtons.YesNo);

var flag = mboxResult == DialogResult.Yes;

Context context;

if (radioButton1.Checked) context = new(new SimpleSort());

else context = new(new ImprovedSort());

context.SortArr(flag);

AddLine();

PrintInfo();

FileOut.sorted = true;

FillLine(false);

if (flag) this.SaveAs\_Click();

} else MessageBox.Show("Îøèáêà! Ñíà÷àëà ñãåíåðèðóéòå ìàññèâ èëè îòêðîéòå åãî èç ôàéëà");

}

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

{

string text;

button1.Enabled = true;

var ofd = new OpenFileDialog();

ofd.Filter = "Text files(\*.txt)|\*.txt";

if (ofd.ShowDialog() == DialogResult.OK)

using (var reader = new StreamReader(ofd.FileName))

text = reader.ReadToEnd();

else return;

var stringArr = text.Split(' ');

var intArr = new int[stringArr.Length];

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

if (int.TryParse(stringArr[i], out intArr[i]) == false)

{

MessageBox.Show("Îøèáêà");

return;

}

Context.array = intArr;

FillLine();

}

private void SaveAs\_Click(object sender=null, EventArgs e=null)

{

if (FileOut.fileString == null)

{

MessageBox.Show("Îøèáêà");

return;

}

var sfd = new SaveFileDialog();

sfd.Filter = "Text files(\*.txt)|\*.txt";

if (sfd.ShowDialog() == DialogResult.OK) FileOut.SaveFile(sfd.FileName);

}

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

{

Form2 form2 = new Form2();

this.Hide();

form2.ShowDialog();

this.Show();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text != "")

{

if (int.TryParse(textBox1.Text, out int value))

{

if (value >= trackBar1.Minimum && value <= trackBar1.Maximum) trackBar1.Value = value;

else MessageBox.Show("Îøèáêà! Ââåäåíî çíà÷åíèå, âûõîäÿùåå çà ïðåäåë 1-40");

}

else MessageBox.Show("Îøèáêà! Íåâåðíûé ôîðìàò ââåäåííûõ äàííûõ äëÿ ãåíåðàöèè!");

}

}

}

}

**Context.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

class Context

{

public static int[] array;

public IStrategy strategy;

public Context(IStrategy strategy)

{

this.strategy = strategy;

}

public void SortArr(bool flag)

{

strategy.SortArr(array, flag);

}

public void SortArr(AnalInfo sort)

{

strategy.AnalSortArr(array, sort);

}

}

}

**FileOut.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

internal class FileOut

{

public static bool sorted = false;

public static string fileString;

public static void Fill()

{

foreach (var i in Context.array)

fileString += i + " ";

fileString += "\n";

}

public static void SaveFile(string path)

{

if (fileString != null && sorted)

{

using (var writer = new StreamWriter(path))

writer.Write(fileString);

}

else

MessageBox.Show("Ошибка");

}

}

}

**SimpleSort.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

internal class SimpleSort : IStrategy

{

private static int comparisons = 0;

private static int permutations = 0;

private static string time = "MM:SS.MS";

public void SortArr(int[] mas, bool flag)

{

System.Diagnostics.Stopwatch myStopwatch = new System.Diagnostics.Stopwatch();

myStopwatch.Start();

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

{

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

{

if (Comparisons(mas, i, j, flag)) Swap(mas, i, j, flag);

}

}

myStopwatch.Stop();

var resultTime = myStopwatch.Elapsed;

time = string.Format("{0:00}:{1:00}.{2:000}", resultTime.Minutes, resultTime.Seconds, resultTime.Milliseconds);

if(flag == false)

{

FileOut.fileString = null;

}

Form1.ReadInfo(comparisons, permutations, time);

comparisons = 0;

permutations = 0;

}

private bool Comparisons(int[] arr, int ind1, int ind2, bool flag)

{

comparisons++;

if (flag && FileOut.fileString == null)

{

FileOut.fileString = "Исходный массив: ";

FileOut.Fill();

}

if (flag)

FileOut.fileString += $"Сравнение {comparisons}: " + $"{arr[ind1]} и {arr[ind2]}\n";

return arr[ind1] > arr[ind2];

}

private void Swap(int[] arr, int ind1, int ind2, bool flag)

{

permutations++;

if (flag) FileOut.fileString += $"Перемещение {permutations}: " + $"[{ind1 + 1}] - {arr[ind1]} и [{ind2 + 1}] - {arr[ind2]}\n";

string swapString = "";

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

if (i == ind1 || i == ind2) swapString += $"[{arr[i]}] ";

else swapString += arr[i] + " ";

Form1.AddSortLine(swapString);

int temp = arr[ind1];

arr[ind1] = arr[ind2];

arr[ind2] = temp;

FileOut.Fill();

}

public void AnalSortArr(int[] arr, AnalInfo sort)

{

System.Diagnostics.Stopwatch myStopwatch = new System.Diagnostics.Stopwatch();

myStopwatch.Start();

int temp;

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

{

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

{

comparisons++;

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

{

permutations++;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

myStopwatch.Stop();

var resultTime = myStopwatch.Elapsed;

sort.time = string.Format("{0:00}:{1:00}.{2:000}", resultTime.Minutes, resultTime.Seconds, resultTime.Milliseconds);

sort.comparisons = comparisons;

sort.permutations = permutations;

comparisons = 0;

permutations = 0;

Form1.line = "";

}

}

}

**AnalInfo.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

public class AnalInfo

{

public int count = 0;

public int comparisons = 0;

public int permutations = 0;

public string time = "";

}

}

**IStrategy.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WinFormsApp1

{

interface IStrategy

{

void SortArr(int[] arr, bool flag);

void AnalSortArr(int[] arr, AnalInfo sort);

}

}