

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет _____ ИТР _____

Кафедра _____ ПИН _____

ЛАБОРАТОРНАЯ РАБОТА №1

По Структуры и алгоритмы обработки данных

Руководитель

Привезенцев Д.Г.

(фамилия, инициалы)

(подпись)

(дата)

Студент _____ ПИН - 121 _____

(группа)

Ермилов М.В.

(фамилия, инициалы)

(подпись)

(дата)

Муром 2023

Лабораторная работа №1

Тема: Алгоритмы сортировки линейных коллекций данных

Ход работы:

1. Реализовать три метода сортировки коллекций согласно таблице 1.
2. Реализовать управляющую программу(ы), включающую:
 - a. ввод исходных данных: из файла, генерацией случайных чисел (способа ввода данных предусмотреть в программе путем ввода выбора пользователя, при этом размер массива тоже указывается во время выполнения программы);
 - b. ввод на экран исходных данных;
 - c. выбор направления сортировки;
 - d. сортировку коллекции;
 - e. вывод на экран результата работы;
 - f. замер времени выполнения сортировки.
3. Реализовать вывод отладочной информации:
 - a. количество сравнений двух элементов;
 - b. количество перестановок двух элементов.
4. Выполнить исследование реализованных трех алгоритмах на разных коллекциях:
 - a. Выполнить замеры времени для коллекций размера 10, 100, 1000, 10000, 100000, 1000000 элементов.
 - b. Заполнить таблицы 2-4 для каждого анализируемого алгоритма.
 - c. Построить графики зависимости времени сортировки, числа перестановок, числа сравнений элементов от размера коллекции для каждого анализируемого алгоритма.
 - d. Сделать выводы по таблицам и графику.

					МИ ВлГУ 09.03.04			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Ермилов М.В.						
Провер.		Привезенцев Д.Г.						
Реценз.								
Н. Контр.								
Утверд.								
						Лит.	Лист	Листов
							2	13
						ПИН-121		

Алгоритмы по заданию:

1. Гномья сортировка
2. Сортировка пузырьком
3. Сортировка подсчетом

Таблица 1 - время выполнения сортировок

	10	100	1 000	10 000	100 000	1 000 000
Алгоритм 1	0 с.	0 с.	0.006 с.	0.559 с.	38.999 с.	1 ч. 6 мин 11.387 с.
Алгоритм 2	0 с.	0 с.	0.029 с.	1.241 с.	1 мин. 34.526 с.	2 ч. 29 мин. 53.955 с.
Алгоритм 3	0.39 с.	0.39 с.	0.029 с.	0.03 с.	0.032 с.	0.102 с.

Таблица 2 - число сравнений значений элементов

	10	100	1 000	10 000	100 000	1 000 000
Алгоритм 1	29	2675	248203	25101803	2499149361	250160341581
Алгоритм 2	90	9900	999000	99990000	9999900000	999999000000
Алгоритм 3	—	—	—	—	—	—

Таблица 3 - число перестановок элементов

	10	100	1 000	10 000	100 000	1 000 000
Алгоритм 1	22	2581	247211	25091814	2499049375	250159341596
Алгоритм 2	24	2383	249455	24967821	2497494415	250085911198
Алгоритм 3	10	100	1000	10000	100000	1000000

Код для быстрой сортировки из консоли:

```
using System;
using System.Collections;
using System.Diagnostics;
using static System.Runtime.InteropServices.JavaScript.JSType;

Random rnd = new Random(111);
Start();

void Start()
{
    for(int j = 0; j < 3; j++)
    {
        switch (j)
```

```

{
    case 0:
        Console.WriteLine("GnomeSort");
        break;
    case 1:
        Console.WriteLine("BubleSort");
        break;
    case 2:
        Console.WriteLine("CountingSort");
        break;
}
for (int i = 10; i <= 1000000; i *= 10)
{
    Console.WriteLine($"\\ni = {i}");
    Console.WriteLine("-----");
    Console.WriteLine("          |          Время          |          Сравнений          |");
    Console.WriteLine("-----");
    Console.WriteLine("Перестановок |");
    Console.WriteLine("-----");
    long timeMin = 0;
    long timeMax = 0;
    double timeAverage = 0;
    long comparisonsMin = 0;
    long comparisonsMax = 0;
    double comparisonsAverage = 0;
    long permutationsMin = 0;
    long permutationsMax = 0;
    double permutationsAverage = 0;
    for (int I = 0; I < 5; I++)
    {
        Stopwatch stopwatch = new Stopwatch();
        int[] array = Generation(i, -1000000, 1000000);
        int[] arrayCopy = Copy(array);
        stopwatch.Start();
        switch (j)
        {
            case 0:
                GnomeSort(array);
                break;
            case 1:
                BubleSort(array);
                break;
            case 2:
                CountingSort(array);
                break;
        }
        stopwatch.Stop();
        long comparisons = 0;
        long permutations = 0;
        switch (j)
        {
            case 0:
                _GnomeSort(arrayCopy, ref comparisons, ref permutations);
                break;
            case 1:
                _BubleSort(arrayCopy, ref comparisons, ref permutations);
                break;
            case 2:
                _CountingSort(arrayCopy, ref comparisons, ref permutations);
                break;
        }
        if(I==0)
        {
            info("Значение", stopwatch.ElapsedMilliseconds, comparisons,
permutations);

```

```

        comparisonsMin = comparisons;
        permutationsMin = permutations;
        timeMin = stopwatch.ElapsedMilliseconds;
    }
    if(timeMin > stopwatch.ElapsedMilliseconds)
        timeMin= stopwatch.ElapsedMilliseconds;
    if(timeMax < stopwatch.ElapsedMilliseconds)
        timeMax= stopwatch.ElapsedMilliseconds;
    timeAverage += stopwatch.ElapsedMilliseconds / 5.0;

    if (comparisonsMin > comparisons)
        comparisonsMin = comparisons;
    if (comparisonsMax < comparisons)
        comparisonsMax = comparisons;
    comparisonsAverage += comparisons / 5.0;

    if (permutationsMin > permutations)
        permutationsMin = permutations;
    if (permutationsMax < permutations)
        permutationsMax = permutations;
    permutationsAverage += permutations / 5.0;
}
info("Минимум", timeMin, comparisonsMin, permutationsMin);
info("Максимум", timeMax, comparisonsMax, permutationsMax);
info("Среднее", (long)timeAverage, (long)comparisonsAverage,
(long)permutationsAverage);
}
Console.WriteLine("\n\n\n\n\n\n\n\n\n\n\n");
}
}
void info(string text, long Milliseconds, long comparisons, long permutations)
{
    Console.WriteLine(System.String.Format("{0,9}|{1,19}|{2,19}|{3,19}|", text,
Milliseconds / 1000.0, comparisons, permutations));
    Console.WriteLine("-----");
    -----");
}
int[] Generation(int Length, int min, int max)
{
    int[] array = new int[Length];
    for(int i = 0; i < Length; i++)
        array[i] = rnd.Next(min, max);
    return array;
}
int[] Copy(int[] array)
{
    int[] arrayCopy = new int[array.Length];
    for(int i = 0; i < array.Length; i++)
        arrayCopy[i] = array[i];
    return arrayCopy;
}
void Swap(ref int A, ref int B)
{
    int C = A;
    A = B;
    B = C;
}
void _GnomeSort(int[] array, ref long comparisons, ref long permutations)
{
    int i = 1;
    int j = 2;
    while (i < array.Length)
    {
        comparisons++;
        if (i > 0 && array[i] > array[i - 1])
        {

```

```

        i = j;
        j++;
    }
    else
    {
        permutations++;
        Swap(ref array[i], ref array[i - 1]);
        i--;
        if (i == 0)
        {
            i = j;
            j++;
        }
    }
}

}

void _BubleSort(int[] array, ref long comparisons, ref long permutations)
{
    for (int k = 0; k < array.Length; k++)
    {
        for (int i = 0; i < array.Length - 1; i++)
        {
            comparisons++;
            if (array[i] > array[i + 1])
            {
                permutations++;
                Swap(ref array[i + 1], ref array[i]);
            }
        }
    }
}

void _CountingSort(int[] array, ref long comparisons, ref long permutations)
{
    int maxV = array.Max();
    int minV = array.Min();
    int[] B = new int[maxV - minV + 1];
    for(int i = 0; i < array.Length; i++)
    {
        B[array[i] - minV]++;
    }
    int q = 0;
    for (int i = 0; i < (maxV - minV + 1); ++i)
    {
        for (int j = 0; j < B[i]; ++j)
        {
            permutations++;
            array[q++] = minV + i;
        }
    }
}

void GnomeSort(int[] array)
{
    int i = 1;
    int j = 2;
    while (i < array.Length)
    {
        if (i > 0 && array[i] > array[i - 1])
        {
            i = j;
            j++;
        }
        else
        {
            Swap(ref array[i], ref array[i - 1]);
            i--;
            if (i == 0)

```

```

        {
            i = j;
            j++;
        }
    }
}

void BubleSort(int[] array)
{
    for (int k = 0; k < array.Length; k++)
    {
        for (int i = 0; i < array.Length - 1; i++)
        {
            if (array[i] > array[i + 1])
            {
                Swap(ref array[i + 1], ref array[i]);
            }
        }
    }
}

void CountingSort(int[] array)
{
    int maxV = array.Max();
    int minV = array.Min();
    int[] B = new int[maxV - minV + 1];
    for (int i = 0; i < array.Length; i++)
    {
        B[array[i] - minV]++;
    }
    int q = 0;
    for (int i = 0; i < (maxV - minV + 1); ++i)
    {
        for (int j = 0; j < B[i]; ++j)
        {
            array[q++] = minV + i;
        }
    }
}

```

Код приложения для генерации и сортировки:

```

namespace app
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI settings or
            // default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new FormMain());
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace app

```

					МИ ВлГУ 09.03.04	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    internal class ArrayAlgorithms
    {
        static private Random rnd = new Random(111);
        static private void Swap(ref int A, ref int B)
        {
            int C = A;
            A = B;
            B = C;
        }
        static public int[] Copy(int[] array)
        {
            int[] arrayCopy = new int[array.Length];
            for (int i = 0; i < array.Length; i++)
                arrayCopy[i] = array[i];
            return arrayCopy;
        }

        static public int[] Generation(int Length, int min, int max)
        {
            int[] array = new int[Length];
            for (int i = 0; i < Length; i++)
                array[i] = rnd.Next(min, max);
            return array;
        }

        static public void GnomeSort(int[] array)
        {
            int i = 1;
            int j = 2;
            while (i < array.Length)
            {
                if (i > 0 && array[i] > array[i - 1])
                {
                    i = j;
                    j++;
                }
                else
                {
                    Swap(ref array[i], ref array[i - 1]);
                    i--;
                    if (i == 0)
                    {
                        i = j;
                        j++;
                    }
                }
            }
        }

        static public void GnomeSort(int[] array, ref long comparisons, ref long
permutations)
        {
            int i = 1;
            int j = 2;
            while (i < array.Length)
            {
                comparisons++;
                if (i > 0 && array[i] > array[i - 1])
                {
                    i = j;
                    j++;
                }
                else
                {
                    permutations++;
                }
            }
        }
    }
}

```



```

        Swap(ref array[i], ref array[i - 1]);
        i--;
        if (i == 0)
        {
            i = j;
            j++;
        }
    }
}

static public void BubleSort(int[] array, ref long comparisons, ref long
permutations)
{
    for (int k = 0; k < array.Length; k++)
    {
        for (int i = 0; i < array.Length - 1; i++)
        {
            comparisons++;
            if (array[i] > array[i + 1])
            {
                permutations++;
                Swap(ref array[i + 1], ref array[i]);
            }
        }
    }
}

static public void BubleSort(int[] array)
{
    for (int k = 0; k < array.Length; k++)
    {
        for (int i = 0; i < array.Length - 1; i++)
        {
            if (array[i] > array[i + 1])
            {
                Swap(ref array[i + 1], ref array[i]);
            }
        }
    }
}

static public void CountingSort(int[] array, ref long comparisons, ref long
permutations)
{
    int maxV = array.Max();
    int minV = array.Min();
    int[] B = new int[maxV - minV + 1];
    for (int i = 0; i < array.Length; i++)
    {
        B[array[i] - minV]++;
    }
    int q = 0;
    for (int i = 0; i < (maxV - minV + 1); ++i)
    {
        for (int j = 0; j < B[i]; ++j)
        {
            permutations++;
            array[q++] = minV + i;
        }
    }
}

static public void CountingSort(int[] array)
{
    int maxV = array.Max();
    int minV = array.Min();
    int[] B = new int[maxV - minV + 1];
    for (int i = 0; i < array.Length; i++)
    {

```

```

        B[array[i] - minV]++;
    }
    int q = 0;
    for (int i = 0; i < (maxV - minV + 1); ++i)
    {
        for (int j = 0; j < B[i]; ++j)
        {
            array[q++] = minV + i;
        }
    }
}

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

namespace app
{
    public partial class FormMain : Form
    {
        int[] array = null;
        int[] arraySort = null;
        public FormMain()
        {
            InitializeComponent();
        }
        private void buttonOpenFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "Файл .json (*.json)|*.json";
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                StreamReader r = new StreamReader(dialog.FileName);
                string strjson = r.ReadToEnd();
                JObject json = JObject.Parse(strjson);
                int Length = (int)json["Length"];
                array = new int[Length];
                for(int i = 0; i < Length; i++)
                    array[i] = (int)json["Array"][i];
                richTextBoxArray.Text = ArrayToText(array);
            }
        }
        private void buttonGeneration_Click(object sender, EventArgs e)
        {
            int l = 0;
            int.TryParse(textBoxSize.Text, out l);
            array = ArrayAlgorithms.Generation(l, -1000000, 1000000);
            richTextBoxArray.Text = ArrayToText(array);
        }
        private void buttonSort_Click(object sender, EventArgs e)
        {
            Stopwatch stopwatch = new Stopwatch();
            long comparisons = 0;
            long permutations = 0;
            if (array != null)
            {

```

```

        arraySort = ArrayAlgorithms.Copy(array);
        if (checkBoxTime.Checked)
            stopwatch.Start();
        if (radioButtonGnomeSort.Checked)
            if (checkBoxData.Checked)
                ArrayAlgorithms.GnomeSort(arraySort, ref comparisons, ref
permutations);
            else
                ArrayAlgorithms.GnomeSort(arraySort);
        else if (radioButtonBubleSort.Checked)
            if (checkBoxData.Checked)
                ArrayAlgorithms.BubleSort(arraySort, ref comparisons, ref
permutations);
            else
                ArrayAlgorithms.BubleSort(arraySort);
        else if (radioButtonCountingSort.Checked)
            if (checkBoxData.Checked)
                ArrayAlgorithms.CountingSort(arraySort, ref comparisons, ref
permutations);
            else
                ArrayAlgorithms.CountingSort(arraySort);
        else
        {
        }
    }
    string r = "";
    if (checkBoxTime.Checked)
    {
        stopwatch.Stop();
        r += $"Время {stopwatch.ElapsedMilliseconds / 1000.0}\n";
    }
    if (checkBoxData.Checked)
        r += $"Сравнений {comparisons}\nПерестановок {permutations}\n";
    richTextBoxSortArray.Text = r + ArrayToText(arraySort);
}
private void buttonSaveSort_Click(object sender, EventArgs e) =>
Save(arraySort);
private void buttonSave_Click(object sender, EventArgs e) => Save(array);
private void Save(int[] array)
{
    SaveFileDialog dialog = new SaveFileDialog();
    dialog.Filter = "файл .json (*.json)|*.json";
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        using(StreamWriter s = File.CreateText(dialog.FileName))
        {
            s.Write(ArrayToJson(array));
        }
    }
}
private string ArrayToText(int[] array)
{
    string r = "";
    for(int i = 0; i < array.Length; i++)
        if(i + 1 != array.Length)
            r += $"{array[i]} ";
        else
            r += $"{array[i]}";
    return r;
}
private string ArrayToJson(int[] array)
{
    string r = "{\n\t\"Length\": " + array.Length + ",\n\t\"Array\":";
    for (int i = 0; i < array.Length; i++)

```

```

        if (i + 1 != array.Length)
            r += $"{array[i]}, ";
        else
            r += $"{array[i]}";
        r += "]\n";
        return r;
    }
}

```

The screenshot shows the 'FormMain' application window. It features a title bar with standard Windows window controls. Below the title bar, there are five buttons: 'Выбрать файл' (Select file), 'Сохранить массив.' (Save array), 'Сохранить сорт.' (Save sort), 'Сгенерировать' (Generate), and a 'Размер' (Size) input field. On the left side, there is a 'Сортировать' (Sort) button and a group of radio buttons for sorting methods: 'Засечь время' (Measure time), 'Дополнительные данные' (Additional data), 'Гномья сортировка' (Gnome sort - selected), 'Сортировка пузырьком' (Bubble sort), and 'Сортировка подсчетом' (Counting sort). To the right of these buttons are two large text input areas labeled 'Массив' (Array) and 'Сортированный массив' (Sorted array).

Рис 1 - пример работы программы

This screenshot shows the 'FormMain' application window with a Windows File Explorer window open over it. The File Explorer window is titled 'Открытие' (Opening) and shows the path 'DPSA > laboratory work 1'. The search bar contains 'Поиск в: laboratory work 1'. The file list shows three items: 'app' (Папка с файлами), 'work' (Папка с файлами), and 'TestSave.json' (JSON File). The 'FormMain' window in the background is partially visible, showing the same interface as in Figure 1, but with the 'Выбрать файл' button highlighted.

Рис 2 - пример работы программы

FormMain

Выбрать файл Сохранить массив. Сохранить сорт. Сгенерировать Размер

Сортировать

☒ Засечь время

☒ Дополнительные данные

☐ Гномья сортировка

☐ Сортировка пузырьком

☒ Сортировка подсчетом

Массив

430906 -226417 -288621 -499348 -358100 123082 280640 -741707 420206 -589848 -862011 -746764 879453 -564307
358056 -295628 44453 630544 363988 -411333

Сортированный массив

Время 0,013
Сравнений 0
Перестановок 20
-862011 -746764 -741707 -589848 -564307 -499348 -411333 -358100 -295628 -288621 -226417 44453 123082 280640
358056 363988 420206 430906 630544 879453

Рис 3 - пример работы программы

Консоль отладки Microsoft Visual Studio

GnomeSort

i = 10

	Время	Сравнений	Перестановок
Значение	0	32	27
Минимум	0	24	15
Максимум	0	33	27
Среднее	0	29	22

i = 100

	Время	Сравнений	Перестановок
Значение	0	2421	2326
Минимум	0	2421	2326
Максимум	0	2767	2674
Среднее	0	2675	2581

i = 1000

	Время	Сравнений	Перестановок
Значение	0,003	242730	241739
Минимум	0,003	242730	241739
Максимум	0,005	255123	254128
Среднее	0,004	248203	247211

Рис 4 - пример работы консольного приложения