ЛАБОРАТОРНАЯ РАБОТА №7

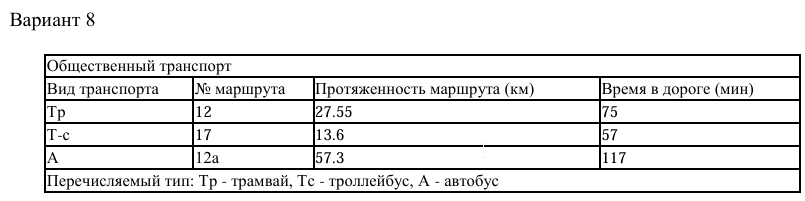
Курс «Алгоритмизация и программирование»

Тема: Простые алгоритмы сортировки. Анализ алгоритмов.

Цель: Научиться программировать простейшие алгоритмы сортировки элементов в массиве; научиться оценивать сложность алгоритмов, усвоить основы О-нотации.

**Ход работы**

**Вариант 8**



1. Дополнить код из лабораторной работы №5 возможностью сортировки записей в таблице по одному из числовых ключей (выберите произвольно). В качестве алгоритма сортировки четным номерам вариантов использовать сортировку выбором, нечетным номерам – вставками.

Новая функция для сортировка таблицы:

// Метод для сортировки таблицы

static void SortTable(ref Transport[] transports, int count)

    {

        // Создаем экземпляр Stopwatch для измерения времени

        Stopwatch stopwatch = new Stopwatch();

        // Запускаем таймер перед сортировкой

        stopwatch.Start();

        Console.WriteLine("\nВыберите столбец для сортировки:");

        Console.WriteLine("1 – По длине маршрута");

        Console.WriteLine("2 – По времени маршрута");

        Console.Write("Введите номер столбца: ");

        int sortChoice = Convert.ToInt32(Console.ReadLine());

        // Сортировка выбором

        for (int i = 0; i < count - 1; i++)

        {

            int minIndex = i;

            for (int j = i + 1; j < count; j++)

            {

                bool condition = false;

                // Условие сортировки по длине маршрута

                if (sortChoice == 1 && transports[j].Length < transports[minIndex].Length)

                {

                    condition = true;

                }

                // Условие сортировки по времени маршрута

                else if (sortChoice == 2 && transports[j].Time < transports[minIndex].Time)

                {

                    condition = true;

                }

                if (condition)

                {

                    minIndex = j;

                }

            }

            // Меняем элементы местами

            if (minIndex != i)

            {

                Transport temp = transports[i];

                transports[i] = transports[minIndex];

                transports[minIndex] = temp;

            }

        }

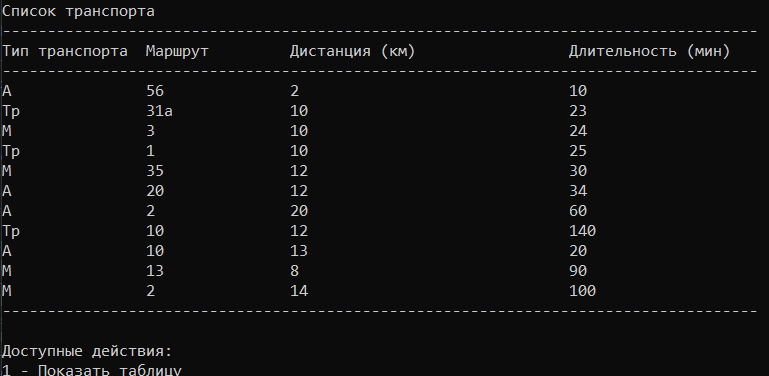
        stopwatch.Stop();

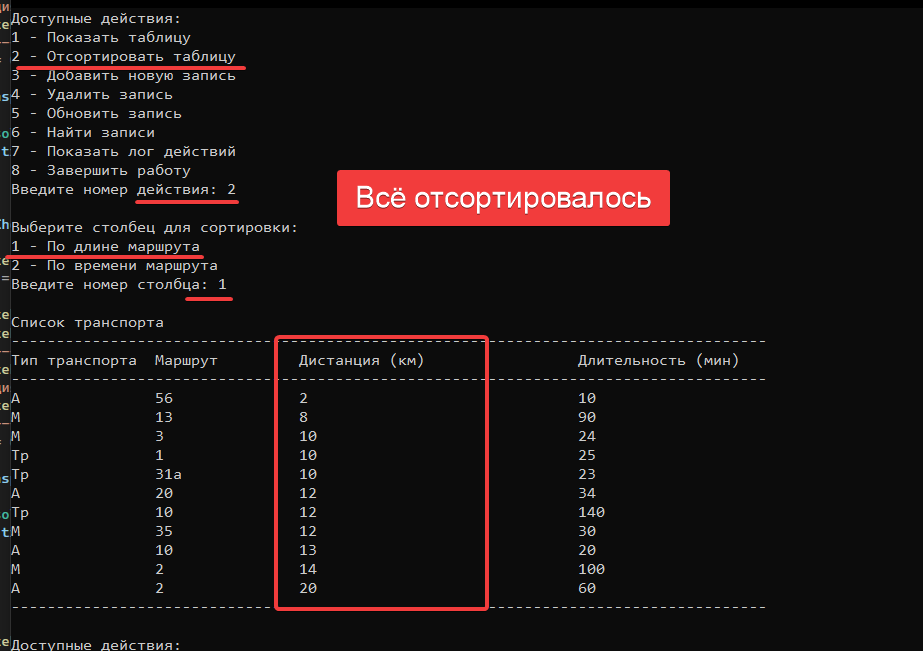
        Console.WriteLine($"\nВремя сортировки: {stopwatch.ElapsedMilliseconds} миллисекунд.");

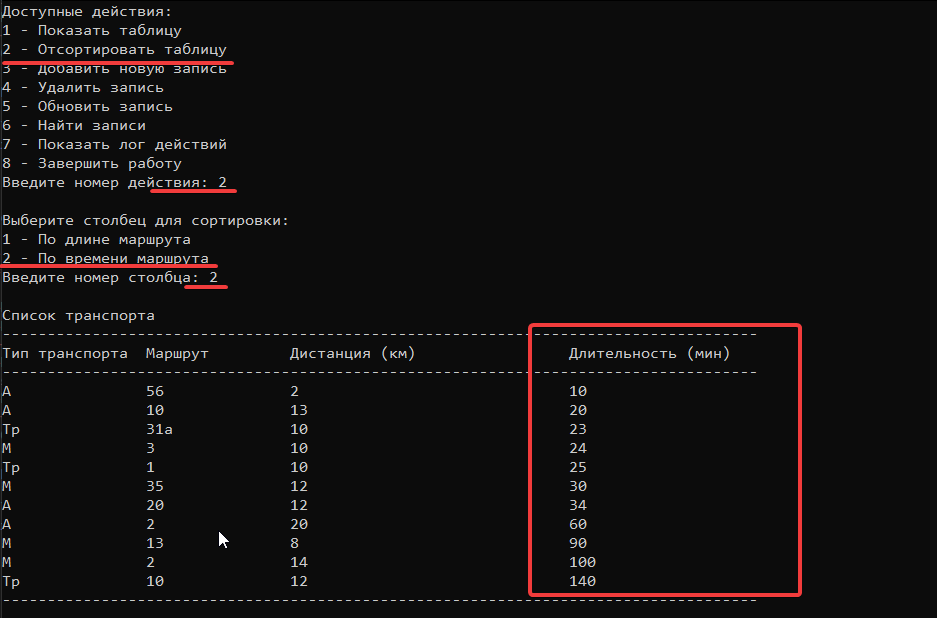
        // После сортировки показываем таблицу

        ViewTable(transports, count);

    }







2. Реализовать в виде отдельных функций алгоритмы сортировки элементов массива (четные номера вариантов – по возрастанию, нечетные номера – по убыванию): выбором, вставками, пузырьком, шейкером, Шелла. Каждую функцию вызвать три раза для разных входных данных:

1) массив из 100 000 элементов типа int, сгенерированный случайным образом;

2) тот же массив, отсортированный в порядке возрастания;

3) тот же массив, отсортированный в порядке убывания.

Вывести на консоль и сравнить время работы всех алгоритмов в каждом случае («секунды : миллисекунды»). Вывести количество сравнений и перестановок элементов для каждого метода сортировки во всех трех случаях. Результаты сортировки программно записать в файл sorted.dat. Написать также код, который считывает данные из этого файла и проверяет, что данные были действительно отсортированы.

using System;

using System.Diagnostics;

using System.IO;

class Program

{

    static void Main()

    {

        // Выполнение сортировки и запись данных в файл

        RunSortingAlgorithm("Сортировка выбором", SelectionSort);

        RunSortingAlgorithm("Сортировка вставками", InsertionSort);

        RunSortingAlgorithm("Сортировка пузырьком", BubbleSort);

        RunSortingAlgorithm("Сортировка шейкером", ShakerSort);

        RunSortingAlgorithm("Сортировка Шелла", ShellSort);

    }

    // Генерация случайного массива

    static int[] GenerateRandomArray(int size)

    {

        Random random = new Random();

        int[] array = new int[size];

        for (int i = 0; i < size; i++)

        {

            array[i] = random.Next();

        }

        return array;

    }

    // Функция для выполнения сортировки и сбора информации о времени и количестве операций

    static void RunSortingAlgorithm(string algorithmName, Action<int[]> sortingAlgorithm)

    {

        int[] originalArray = GenerateRandomArray(100000);

        RunSortingAndWriteToFile(algorithmName, sortingAlgorithm, (int[])originalArray.Clone());

    }

    // Функция для выполнения сортировки, записи результатов в файл и вывод времени

    static void RunSortingAndWriteToFile(string algorithmName, Action<int[]> sortingAlgorithm, int[] array)

    {

        Console.WriteLine($"{algorithmName}. Ждем-с");

        var stopwatch = Stopwatch.StartNew();

        sortingAlgorithm(array);  // Сортируем массив

        stopwatch.Stop();

        Console.WriteLine($"{algorithmName} завершена за: {stopwatch.Elapsed.Seconds} секунд и {stopwatch.Elapsed.Milliseconds} миллисекунд.");

        // Запись отсортированного массива в файл

        string filename = $"{algorithmName}.dat";

        File.WriteAllText(filename, string.Join(",", array));

        // Проверка правильности сортировки

        CheckSortedData(filename);

    }

    // Проверка данных на правильность сортировки

    static void CheckSortedData(string filename)

    {

        string fileContent = File.ReadAllText(filename);

        string[] stringNumbers = fileContent.Split(',');

        int[] numbers = Array.ConvertAll(stringNumbers, s => int.Parse(s));

        bool isSorted = true;

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

        {

            if (numbers[i] < numbers[i - 1])

            {

                isSorted = false;

                break;

            }

        }

        if (isSorted)

        {

            Console.WriteLine("Данные в файле отсортированы правильно.\n");

        }

        else

        {

            Console.WriteLine("Данные в файле НЕ отсортированы.\n");

        }

    }

    // Реализация сортировки выбором

    static void SelectionSort(int[] array)

    {

        long comparisons = 0, swaps = 0;  // Используем long т.к. иногда значения не влазиют в int

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

        {

            int minIndex = i;

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

            {

                comparisons++;

                if (array[j] < array[minIndex])

                {

                    minIndex = j;

                }

            }

            if (minIndex != i)

            {

                swaps++;

                int temp = array[i];

                array[i] = array[minIndex];

                array[minIndex] = temp;

            }

        }

        Console.WriteLine($"Сортировка выбором - Сравнений: {comparisons}, Перестановок: {swaps}");

    }

    // Реализация сортировки вставками

    static void InsertionSort(int[] array)

    {

        long comparisons = 0, swaps = 0;

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

        {

            int key = array[i];

            int j = i - 1;

            while (j >= 0)

            {

                comparisons++;

                if (array[j] > key)

                {

                    array[j + 1] = array[j];

                    j--;

                    swaps++;

                }

                else

                {

                    break;

                }

            }

            array[j + 1] = key; // Перемещение ключа

        }

        Console.WriteLine($"Сортировка вставками - Сравнений: {comparisons}, Перестановок: {swaps}");

    }

    // Реализация сортировки пузырьком

    static void BubbleSort(int[] array)

    {

        long comparisons = 0, swaps = 0;

        bool swapped;

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

        {

            swapped = false;

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

            {

                comparisons++;

                if (array[j] > array[j + 1])

                {

                    swaps++;

                    int temp = array[j];

                    array[j] = array[j + 1];

                    array[j + 1] = temp;

                    swapped = true;

                }

            }

            if (!swapped) break;

        }

        Console.WriteLine($"Сортировка пузырьком - Сравнений: {comparisons}, Перестановок: {swaps}");

    }

    // Реализация сортировки шейкером

    static void ShakerSort(int[] array)

    {

        long comparisons = 0, swaps = 0;

        int left = 0, right = array.Length - 1;

        while (left < right)

        {

            for (int i = left; i < right; i++)

            {

                comparisons++;

                if (array[i] > array[i + 1])

                {

                    swaps++;

                    int temp = array[i];

                    array[i] = array[i + 1];

                    array[i + 1] = temp;

                }

            }

            right--;

            for (int i = right; i > left; i--)

            {

                comparisons++;

                if (array[i] < array[i - 1])

                {

                    swaps++;

                    int temp = array[i];

                    array[i] = array[i - 1];

                    array[i - 1] = temp;

                }

            }

            left++;

        }

        Console.WriteLine($"Сортировка шейкером - Сравнений: {comparisons}, Перестановок: {swaps}");

    }

    // Реализация сортировки Шелла

    static void ShellSort(int[] array)

    {

        long comparisons = 0, swaps = 0;

        int gap = array.Length / 2;

        while (gap > 0)

        {

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

            {

                int temp = array[i];

                int j = i;

                while (j >= gap && array[j - gap] > temp)

                {

                    comparisons++; // Сравнение

                    array[j] = array[j - gap];

                    j -= gap;

                    swaps++; // Перестановка

                }

                array[j] = temp;

            }

            gap /= 2; // Уменьшаем шаг

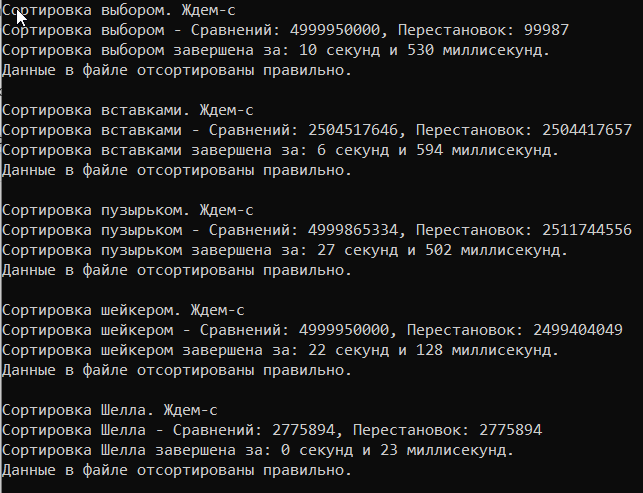
        }

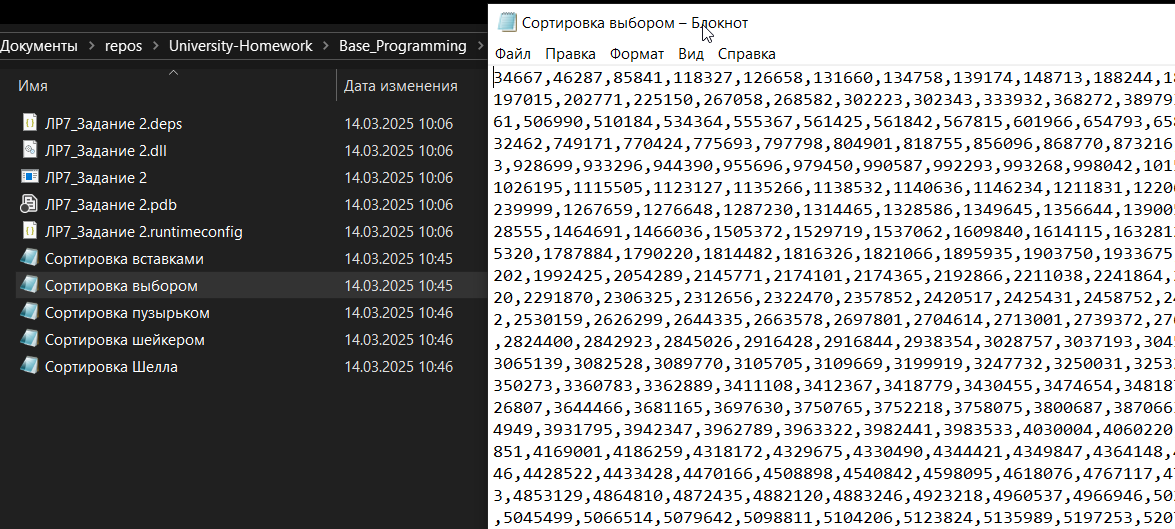
        Console.WriteLine($"Сортировка Шелла - Сравнений: {comparisons}, Перестановок: {swaps}");

    }

}

Программа для каждого алгоритма сортировки сама генерирует числа, сортирует их, замеряет количество сравнений и перестановок, время выполнения сортировок, записывает результаты в файлы и после проверяет расположение чисел в файле по возрастанию:





3. Произвести расчет временной сложности всех алгоритмов, разработанных в лабораторных работах №2, 3 и 4.

Для подсчёта времени работы программ был написан код:

using System;

using System.Diagnostics;

using System.IO;

class Program

{

    static void Main()

    {

        Console.WriteLine("Введите путь к исполняемой программе:");

        string programPath = Console.ReadLine();

        // Проверка, существует ли файл по указанному пути

        if (File.Exists(programPath))

        {

            // Засекаем время выполнения программы

            MeasureExecutionTime(programPath);

        }

        else

        {

            Console.WriteLine("Ошибка: Указанный файл не существует.");

        }

    }

    // Метод для измерения времени выполнения программы

    static void MeasureExecutionTime(string programPath)

    {

        Stopwatch stopwatch = Stopwatch.StartNew();

        // Настройка процесса для запуска программы

        Process process = new Process();

        process.StartInfo.FileName = programPath;

        process.StartInfo.RedirectStandardOutput = true; // Перенаправление вывода программы в консоль

        process.StartInfo.RedirectStandardError = true; // Перенаправление ошибок в консоль

        try

        {

            process.Start();

            process.WaitForExit();

            stopwatch.Stop();

            Console.WriteLine($"Программа завершена. Время выполнения: {stopwatch.ElapsedMilliseconds / 1000} секунд и {stopwatch.ElapsedMilliseconds % 1000} миллисекунд.");

        }

        catch (Exception ex)

        {

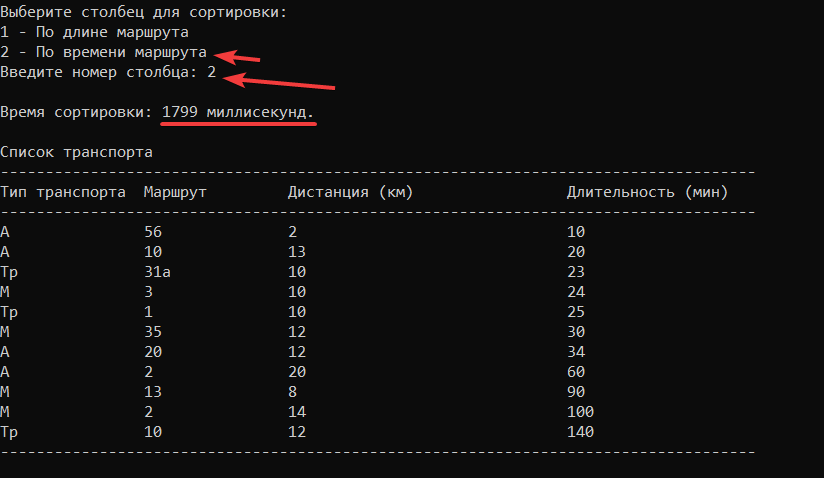
            Console.WriteLine($"Ошибка при запуске программы: {ex.Message}");

        }

    }

}

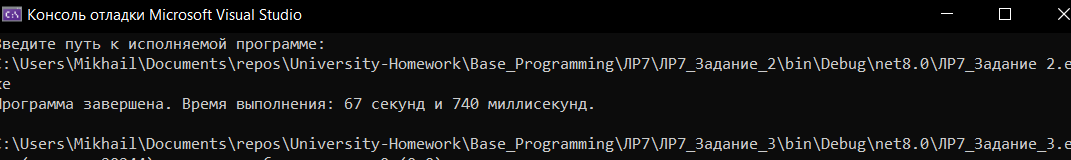
Время выполнения сортировки в Задании 1:

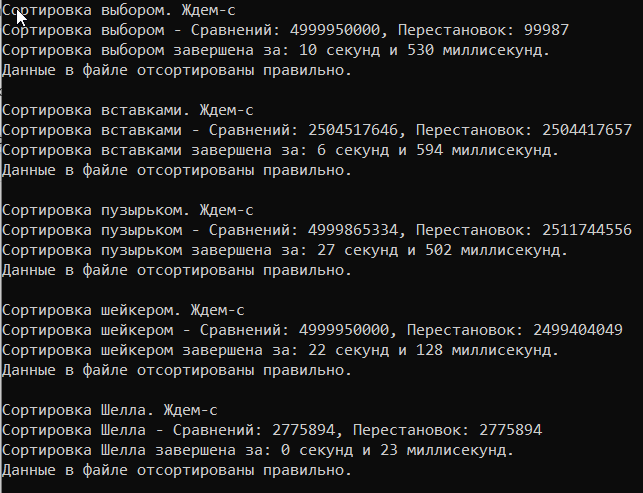


Загрузка данных, сохранение данных и поиск имеют линейную сложность O(n).

Сортировка выбором имеет сложность O(n2) так как для каждой позиции в массиве программа каждый раз находит минимальный элемент.

Время выполнения Задания 2:





Загрузка данных, сохранение данных и поиск имеют линейную сложность O(n). Сортировка выбором имеет сложность O(n²), так как для каждой позиции в массиве программа каждый раз находит минимальный элемент.

Сортировка вставками также имеет сложность O(n²) в худшем случае, так как для каждого элемента нужно сравнивать его с предыдущими элементами.

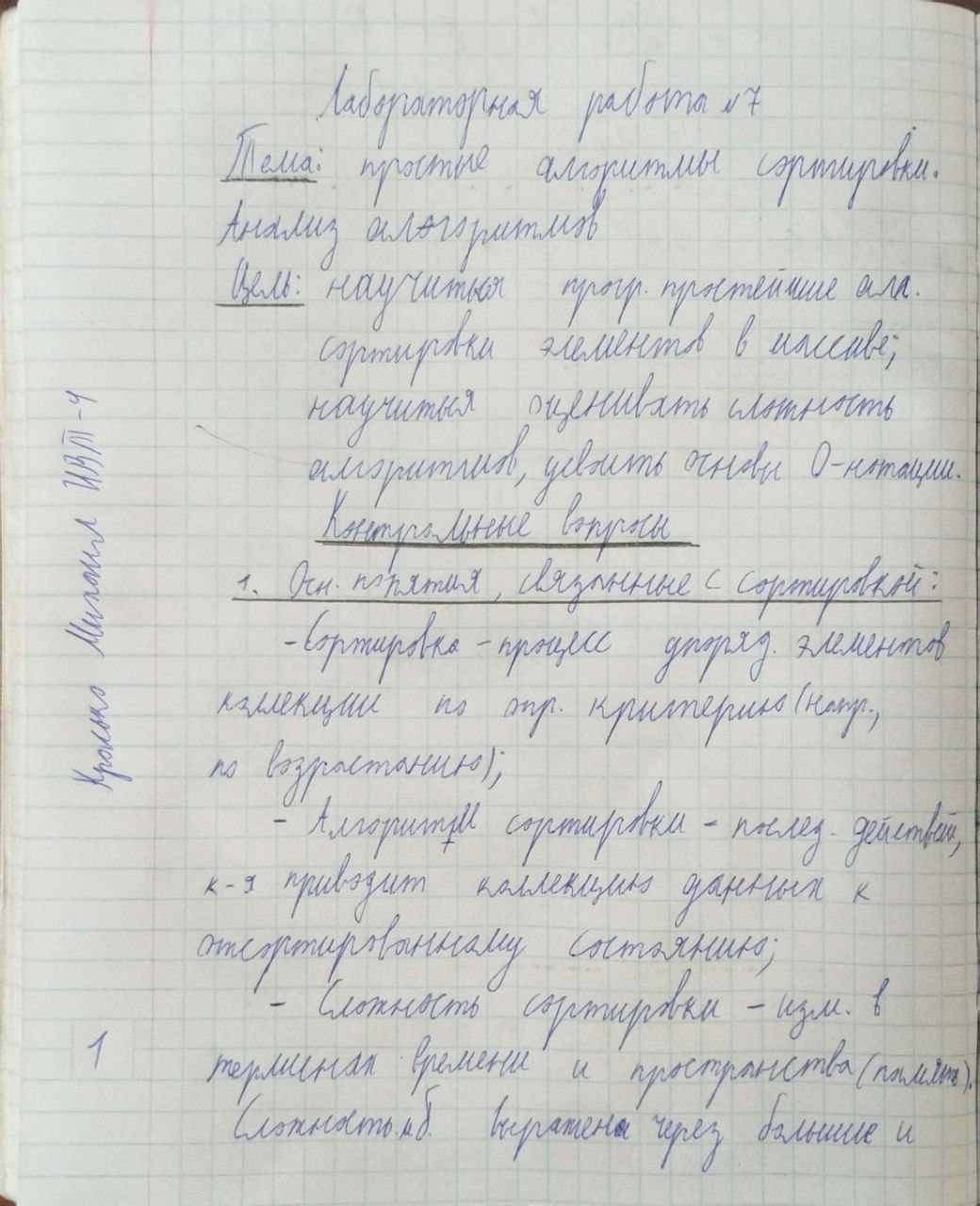
Сортировка пузырьком имеет такую же сложность O(n²), так как алгоритм несколько раз проходит по массиву, сравнивая соседние элементы и меняя их местами, если они в неправильном порядке.

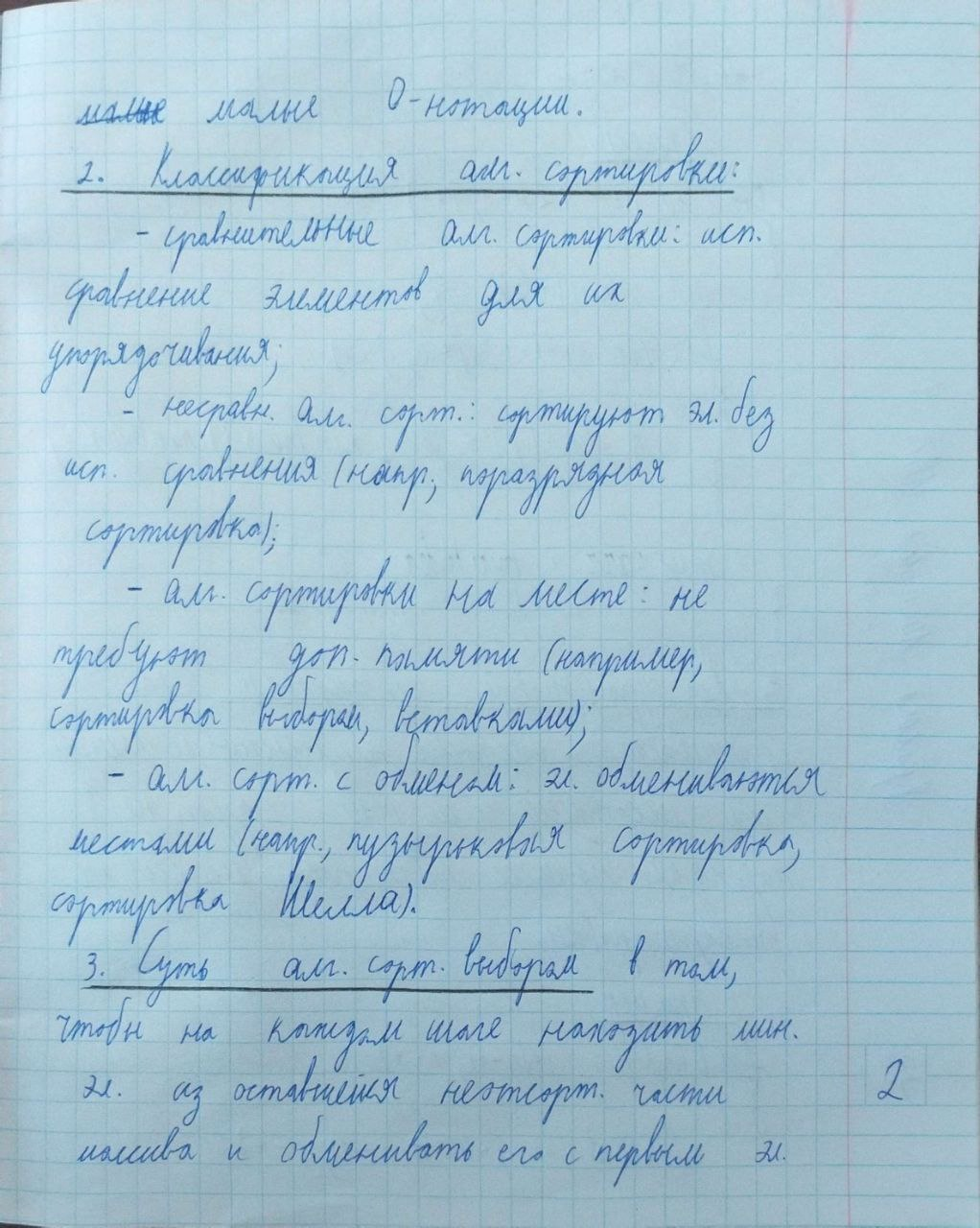
Сортировка шейкером также имеет сложность O(n²). Алгоритм проходит по массиву в обе стороны.

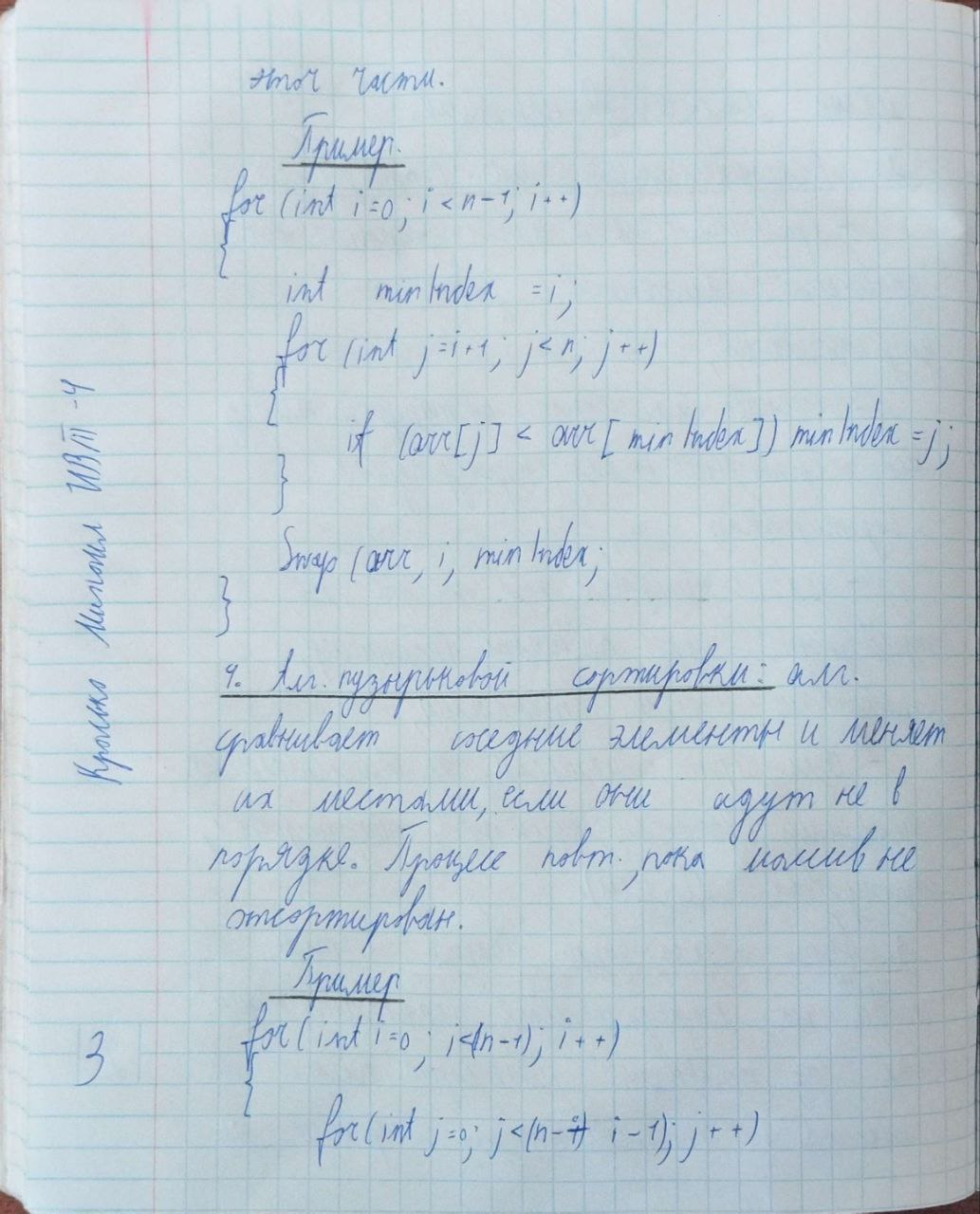
Сортировка Шелла имеет зависимость от шага, однако в худшем случае её сложность также составляет O(n²). В зависимости от того как изменяется шаг эта сортировка может быть быстрее сортировки пузырьком или выбором, но всё равно в худшем случае остаётся квадратичной.

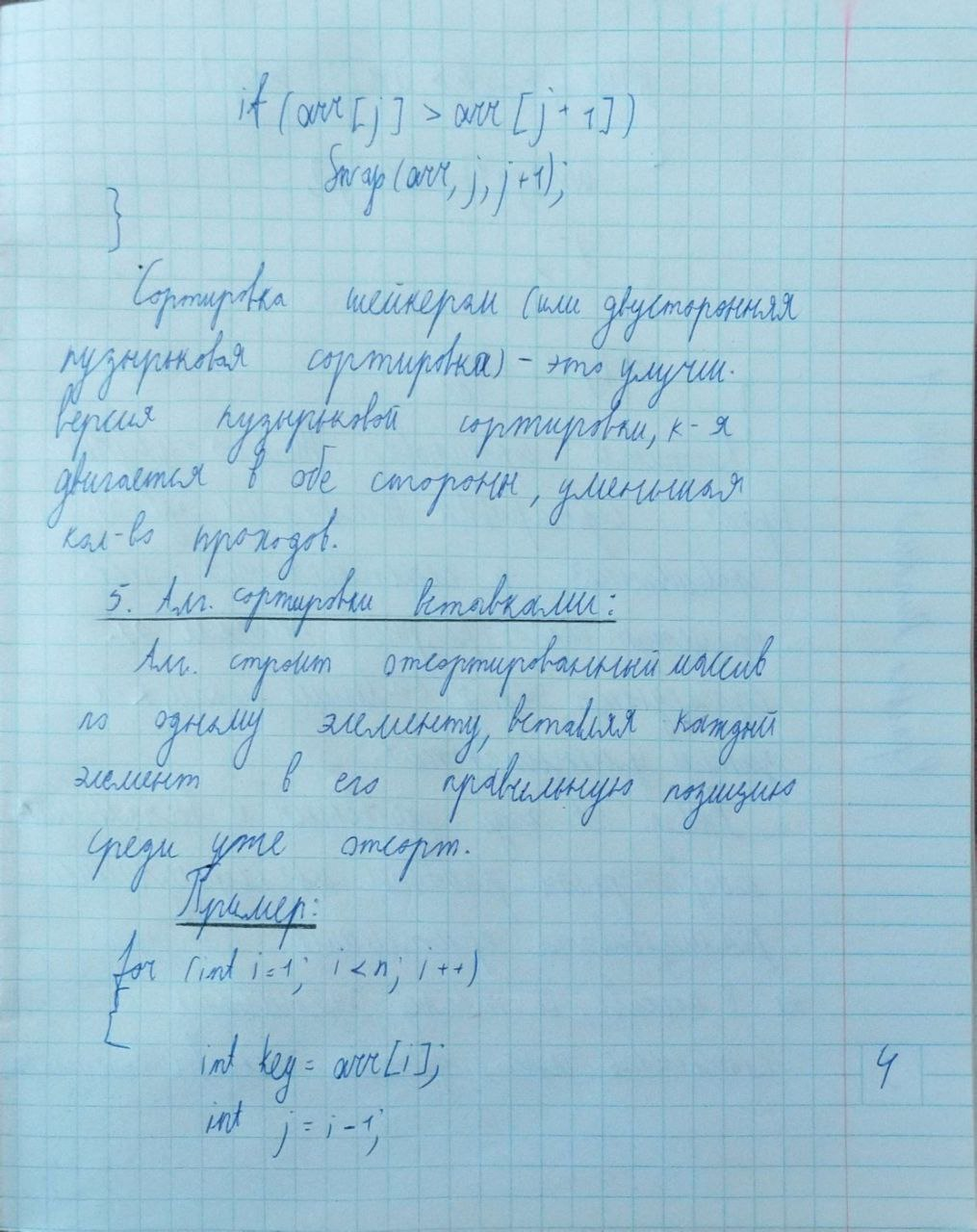
**Контрольные вопросы на следующей странице**

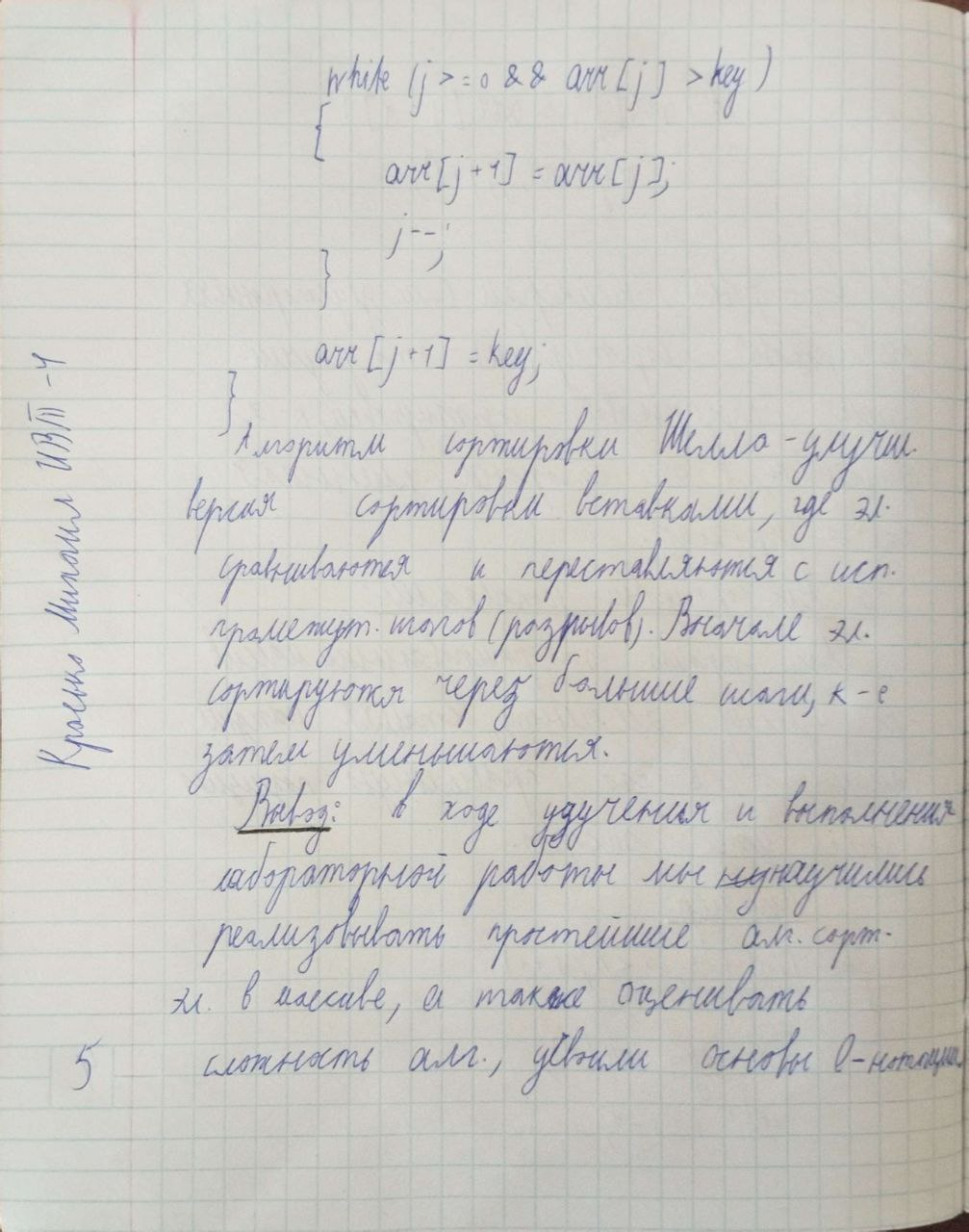
**Контрольные вопросы**











**Вывод:** в ходе изучения и выполнения лабораторной работы мы научились реализовывать простейшие алгоритмы сортировки элементов в массиве, а также оценивать сложность алгоритмов, усвоили основы O-нотации.