Московский авиационный институт

(Национальный исследовательский университет)

Факультет №3

Кафедра 304

Отчёт по лабораторной работе

По учебной дисциплине

Структуры и алгоритмы обработки данных

На тему:  
“Работа с сортировками”

Выполнил:

Плотников Кирилл

Крещук Николай

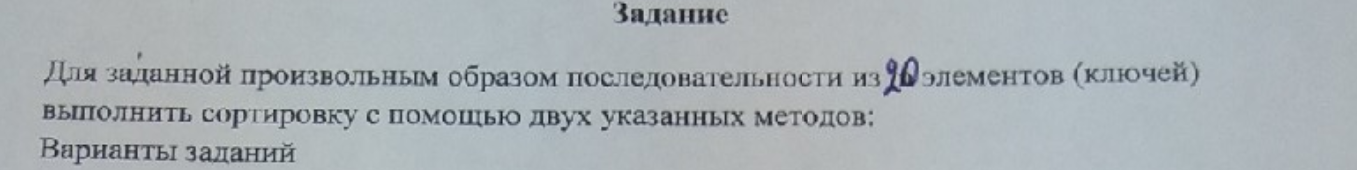
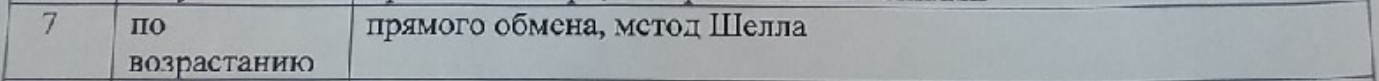
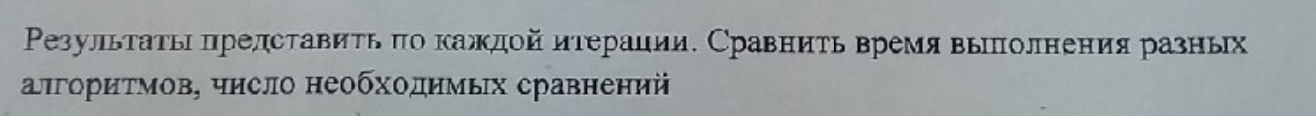
Группа: М30-224БК-17

Бригада №7

Принял:

Силаев А.В.

Москва, 2018

**Класс сортировки Шелла**

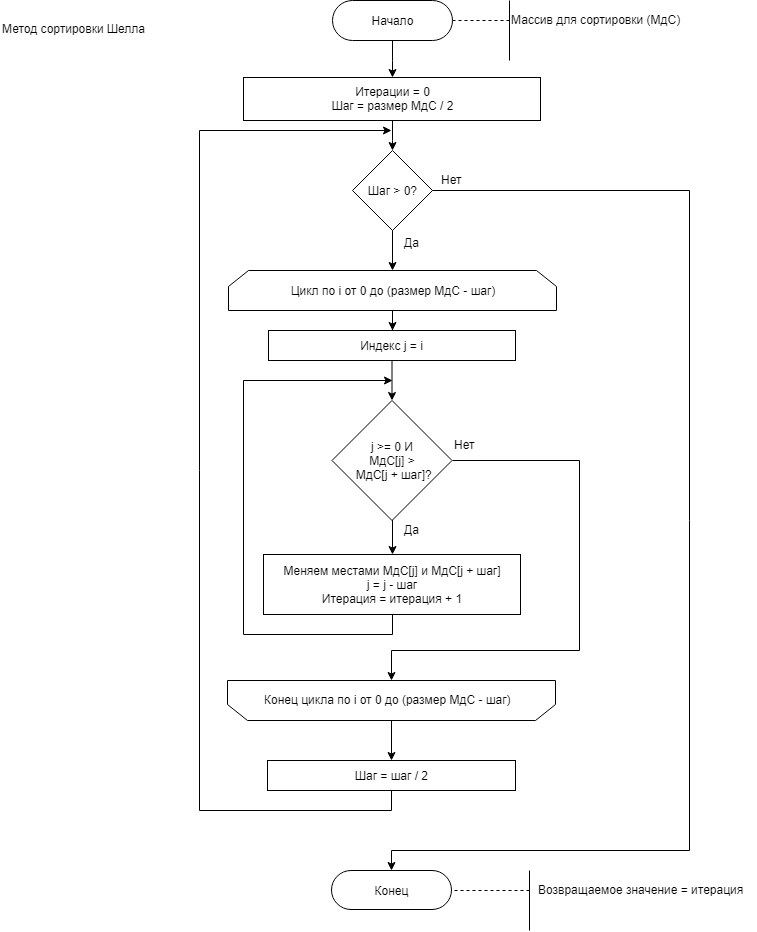
**Имя:** ShellSort.

**Поля:**

1. Имя – строковое значение.

**Методы:**

1. Конструктор – инициализирует новый экземпляр класса и его имя.
2. Сортировка – производит сортировку заданного массива методом Шелла, возвращает количество итераций.



**Класс сортировки пузырьком**

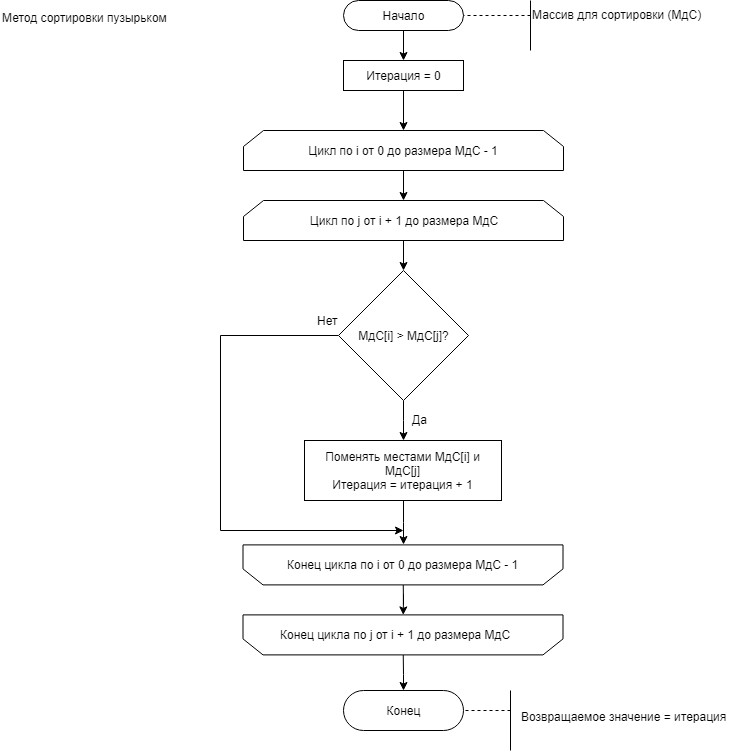
**Имя:** BubbleSort.

**Поля:**

1. Имя – строковое значение.

**Методы:**

1. Конструктор – инициализирует новой экземпляр класса и его имя.
2. Сортировка – сортирует указанный массив методом пузырька и возвращает количество итераций.



**Класс основной программы**

**Методы:**

1. Main – запускает программу, затем инициализирует массив сортировок, в котором их 2: сортировка пузырьком и сортировка Шелла, в итоге вызывает метод DoAlgorithm.
2. DoAlgorithm – инициализирует тестовый массив с помощью метода GetTestArr, инициализирует таймер и вызывает для каждого элемента массива сортировок метод Sort, с последующим выводом результатов.
3. GetTestArr – возвращает случайный массив.
4. PrintArr – печатает массив, который подаётся как входные данные.

**Код программы.**

class Program

{

static void Main(string[] args)

{

ISort[] sorts = new ISort[] { new BubbleSort(), new ShellSort() };

DoAlgorithm(sorts);

Console.ReadLine();

}

static void DoAlgorithm(ISort[] sorts)

{

Console.Write("Введите желаемое кол-во элементов: ");

int wishCountOfElems = int.Parse(Console.ReadLine());

Stopwatch sw = new Stopwatch();

int[] testArr = GetTestArr(wishCountOfElems);

int[] tempArr = new int[testArr.Length];

int iterations = 0;

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

PrintArr(testArr);

foreach (ISort sort in sorts)

{

testArr.CopyTo(tempArr, 0);

sw.Start();

iterations = sort.Sort(tempArr);

sw.Stop();

Console.WriteLine($"Затраченное время на сортировку <{sort.Name}>: {sw.ElapsedMilliseconds} миллисекунд.");

Console.WriteLine($"Кол-во итераций: {iterations}.");

}

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

PrintArr(tempArr);

}

static int[] GetTestArr(int countOfElms)

{

Random rnd = new Random();

int maxValue = countOfElms \* 10000;

int[] testArray = new int[countOfElms];

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

testArray[i] = rnd.Next(-maxValue, maxValue);

return testArray;

}

static void PrintArr(int[] arr)

{

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

Console.WriteLine(arr[i]);

}

}

interface ISort

{

/// <summary>

/// Имя сортировки.

/// </summary>

string Name { get; }

/// <summary>

/// Сортировка массива.

/// </summary>

/// <param name="arr">Массив для сортировки.</param>

/// <returns>Кол-во интераций.</returns>

int Sort(int[] arr);

}

class BubbleSort : ISort

{

/// <summary>

/// Имя сортировки.

/// </summary>

public string Name { get; }

/// <summary>

/// Конструктор по умолчанию класса пузьковой сортировки.

/// </summary>

public BubbleSort()

{

Name = "Сортировка пузьрьковая (прямого обмена)";

}

/// <summary>

/// Сортировка массива метод пузырька.

/// </summary>

/// <param name="arr">Массив для сортировки.</param>

public int Sort(int[] arr)

{

int iterations = 0;

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

{

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

{

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

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

iterations++;

}

}

}

return iterations;

}

}

class ShellSort : ISort

{

/// <summary>

/// Имя сортировки.

/// </summary>

public string Name { get; }

/// <summary>

/// Конструктор по умолчанию класса сортировки Шелла.

/// </summary>

public ShellSort()

{

Name = "Сортировка Шелла";

}

/// <summary>

/// Сортировка массива методов Шелла.

/// </summary>

/// <param name="arr">Массив для сортировки.</param>

/// <returns>Кол-во итераций.</returns>

public int Sort(int[] arr)

{

int iterations = 0;

int j;

int step = arr.Length / 2;

while (step > 0)

{

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

{

j = i;

while ((j >= 0) && (arr[j] > arr[j + step]))

{

int tmp = arr[j];

arr[j] = arr[j + step];

arr[j + step] = tmp;

j -= step;

iterations++;

}

}

step = step / 2;

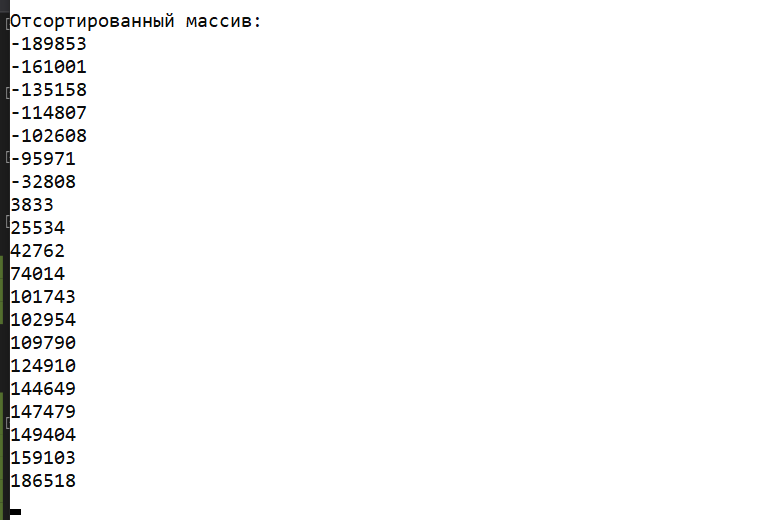
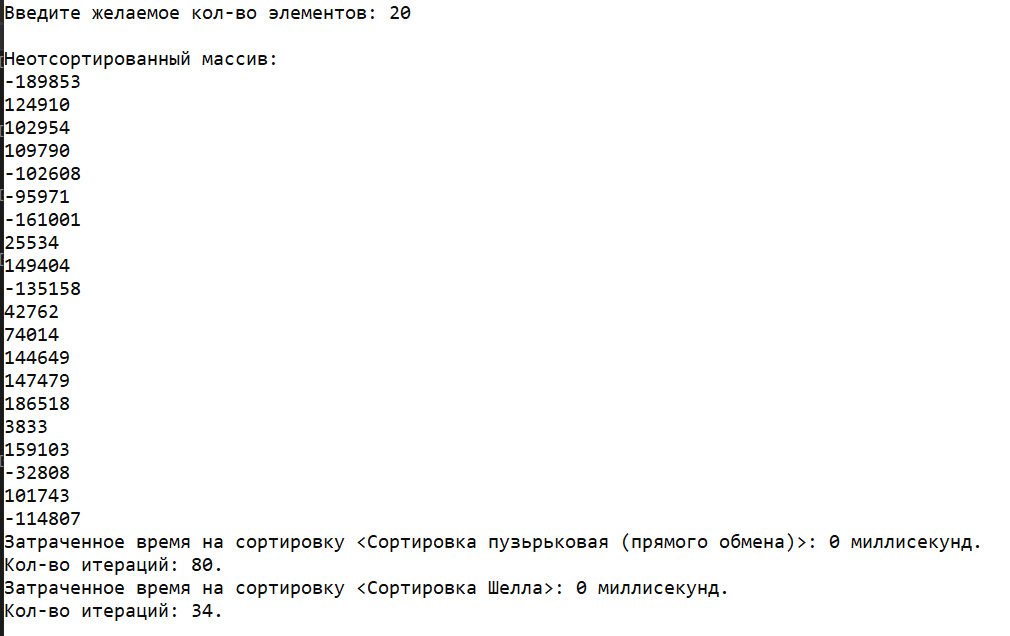
}

return iterations;

}

}

**Результаты работы программы.**



**Выводы.**

Сортировка методом Шелла оказывает в несколько раз быстрее сортировки прямого обмена так как она является улучшенной сортировкой прямого обмена с лучшим временем работы O(n \* log2n), и худшем O(n2), при этом сортировка прямого обмена в среднем имеет время работы O(n2).