

Лабораторная работа №1  
по дисциплине «Структура и алгоритмы и обработки данных»  
на тему:  
«Методы сортировки»

Выполнили: студ. гр. БСТ1902

Козлов М. С.

Вариант №7

Москва 2020

## 1. Ход выполнения лабораторной работы

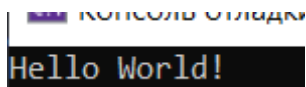
### 1.1 Задание 1

Вывести на консоль "Hello World!"

Код программы:

```
//Задание 1
Console.WriteLine("Hello World!");
```

Вывод программы:

A screenshot of a console window with a black background. The text "Hello World!" is displayed in a light blue or cyan monospaced font.

### 1.2 Задание 2

Написать генератор случайных матриц(многомерных), который принимает опциональные параметры m, n, min\_limit, max\_limit, где m и n указывают размер матрицы, а min\_lim и max\_lim - минимальное и максимальное значение для генерируемого числа . По умолчанию при отсутствии параметров принимать следующие значения:

Код программы:

```
public class Matrix
{
    public int[][] Value { get; private set; }

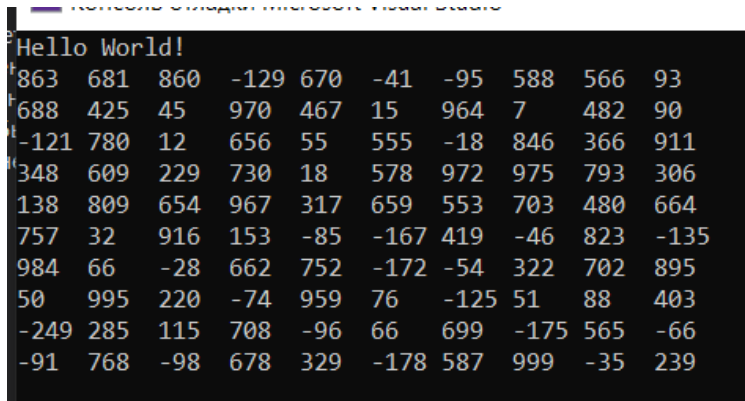
    ...

    public void Create(int variant)
    {
        rnd = new Random();
        maxLimit += variant;

        for (int i = 0; i < Value.Length; i++)
        {
            Value[i] = new int[m];
            for (int j = 0; j < Value[i].Length ; j++)
                Value[i][j] = rnd.Next(minLimit, maxLimit);
        }
    }
}
```

Вызов матрицы:

```
var matrix = new Matrix(10, 10);  
matrix.Create(7);  
Console.WriteLine(matrix.ToString());
```

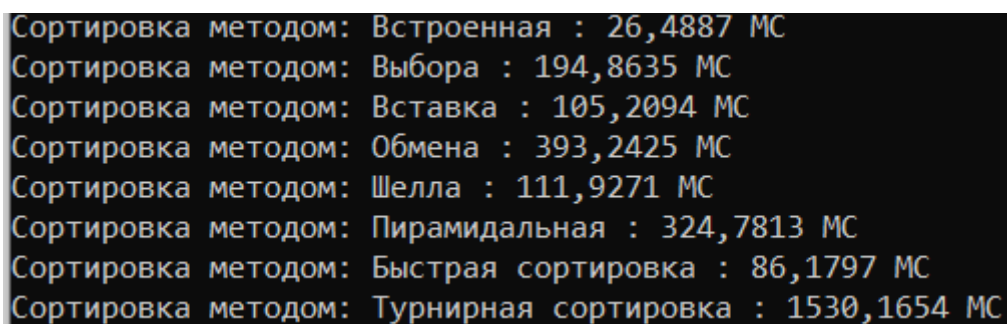


```
Hello World!  
863 681 860 -129 670 -41 -95 588 566 93  
688 425 45 970 467 15 964 7 482 90  
-121 780 12 656 55 555 -18 846 366 911  
348 609 229 730 18 578 972 975 793 306  
138 809 654 967 317 659 553 703 480 664  
757 32 916 153 -85 -167 419 -46 823 -135  
984 66 -28 662 752 -172 -54 322 702 895  
50 995 220 -74 959 76 -125 51 88 403  
-249 285 115 708 -96 66 699 -175 565 -66  
-91 768 -98 678 329 -178 587 999 -35 239
```

### 1.3 Задание 3

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Результаты времени сортировки для матрицы 100x10000



```
Сортировка методом: Встроенная : 26,4887 МС  
Сортировка методом: Выбора : 194,8635 МС  
Сортировка методом: Вставка : 105,2094 МС  
Сортировка методом: Обмена : 393,2425 МС  
Сортировка методом: Шелла : 111,9271 МС  
Сортировка методом: Пирамидальная : 324,7813 МС  
Сортировка методом: Быстрая сортировка : 86,1797 МС  
Сортировка методом: Турнирная сортировка : 1530,1654 МС
```

## Метод сортировки Выбором

```
public static void ChoiceSort(int[] array)
{
    for (int i = 0; i < array.Length - 1; i++)
    {
        int min = i;
        for (int j = i + 1; j < array.Length; j++)
            if (array[j] < array[min])
                min = j;
        Swap(ref array[min], ref array[i]);
    }
}
```

## Метод сортировки Вставкой

```
public static void InsertionSort(int[] array) //n^2
{
    for (int i = 1; i < array.Length; i++)
    {
        int current = array[i];
        int j = i;
        while (j > 0 && current < array[j - 1])
        {
            array[j] = array[j - 1];
            j--;
        }
        array[j] = current;
    }
}
```

## Метод сортировки Обменом

```
public static void ExchangeSort(int[] array)
{
    var lenght = array.Length;
    for (var i = 1; i < lenght; i++)
        for (var j = 0; j < lenght - i; j++)
            if (array[j] > array[j + 1])
                Swap(ref array[j], ref array[j + 1]);
}
```

## Метод сортировки Шелла

```
public static void ShellSort(int[] array)
{
    var d = array.Length / 2;
    while (d >= 1)
    {
        for (var i = d; i < array.Length; i++)
        {
            var j = i;
```

```

        while (j >= d && array[j - d] > array[j])
        {
            Swap(ref array[j], ref array[j - d]);
            j = j - d;
        }
    }
    d = d / 2;
}
}

```

## Метод быстрой сортировки

```

public static void QuickSort(int[] array)
{
    QuickSort(array, 0, array.Length - 1);
}

private static void QuickSort(int[] array, int minIndex, int
maxIndex)
{
    if (minIndex >= maxIndex) return;
    var pivotIndex = Partition(array, minIndex, maxIndex);
    QuickSort(array, minIndex, pivotIndex - 1);
    QuickSort(array, pivotIndex + 1, maxIndex);
}

private static int Partition(int[] array, int minIndex, int maxIndex)
{
    var pivot = minIndex - 1;
    for (var i = minIndex; i < maxIndex; i++)
    {
        if (array[i] < array[maxIndex])
        {
            pivot++;
            Swap(ref array[pivot], ref array[i]);
        }
    }
    pivot++;
    Swap(ref array[pivot], ref array[maxIndex]);

    return pivot;
}

```

## Метод сортировки Пирамидальный

```

public static void PyramidSort(int[] array)
{
    var lenght = array.Length;
    for (int i = lenght / 2 - 1; i >= 0; --i)
    {
        long prev_i = i;
        i = AddToPyramid(array, i, lenght);
        if (prev_i != i) ++i;
    }
    for (int k = lenght - 1; k > 0; --k)
    {
        Swap(ref array[0], ref array[k]);
        int i = 0, prev = -1;
        while (i != prev)
        {

```

```

        prev = i;
        i = AddToPyramid(array, i, k);
    }
}

private static int AddToPyramid(int[] arr, int i, int N)
{
    int iMax;
    if ((2 * i + 2) < N)
    {
        if (arr[2 * i + 1] < arr[2 * i + 2]) iMax = 2 * i + 2;
        else iMax = 2 * i + 1;
    }
    else iMax = 2 * i + 1;
    if (iMax >= N) return i;
    if (arr[i] < arr[iMax])
    {
        Swap(ref arr[i], ref arr[iMax]);
        if (iMax < N / 2) i = iMax;
    }
    return i;
}

```

## Метод турнирной сортировки

```

public static void Tournament(int[] array)
{
    var three = new int[2 * (array.Length + array.Length % 2)][1];
    var index = three.Length - array.Length + array.Length % 2;

    for (int i = index; i < three.Length; i++)
        three[i] = new int[1] { i - index, array[i - index] };

    for (int j = 0; j < array.Length; j++)
    {
        var n = array.Length;
        index = three.Length - array.Length + array.Length % 2;

        while (index > -1)
        {
            n = (n + 1) / 2;

            for (int i = 0; i < n; i++)
            {
                var iCopy = Math.Max(index + i * 2, 1);
                if (three[iCopy] != null && three[iCopy + 1] != null)
                {
                    if (three[iCopy][1] < three[iCopy + 1][1])
                    {
                        three[iCopy / 2] = three[iCopy];
                    }
                    else
                    {
                        three[iCopy / 2] = three[iCopy + 1];
                    }
                }
                else
                {
                    three[iCopy / 2] = (three[iCopy] != null) ?
three[iCopy] : three[iCopy + 1];
                }
            }
            index = iCopy;
        }
    }
}

```

```

        }
    }
    index -= n;
}

index = three[0][0];
var x = three[0][1];
array[j] = x;
three[three.Length - array.Length - array.Length % 2 + index]
= null;
}

```

## 1.4 Задание 4

Создать публичный репозиторий на github