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. **Задание 1**

Вывести на консоль ”Hello World!”

Код программы:

//Задание 1

Console**.**WriteLine**(**"Hello World!"**);**

Вывод программы:



* 1. **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**);**

**}**

**}**

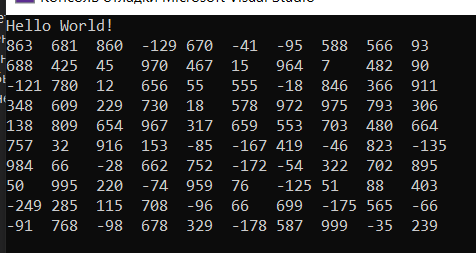
**}**

Вызов матрицы:

var matrix **=** **new** Matrix**(**10**,** 10**);**

matrix**.**Create**(**7**);**

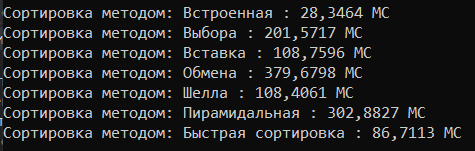
Console**.**WriteLine**(**matrix**.**ToString**());**



* 1. **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 100 x 100000 элементов:



**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**)**

**{**

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

**{**

int cur **=** array**[**i**];**

int j **=** i**;**

**while** **(**j **>** 0 **&&** cur **<** array**[**j **-** 1**])**

**{**

array**[**j**]** **=** array**[**j **-** 1**];**

j**--;**

**}**

array**[**j**]** **=** cur**;**

**}**

**}**

**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**;**

**}**

* 1. **Задание 4**

Создать публичный репозиторий на github