Белорусско-Российский университет

Кафедра ПОИТ

Дисциплина ООПП

Отчет по лабораторной работе №2

«МАССИВЫ.   
ИНДЕКСАТОРЫ.»

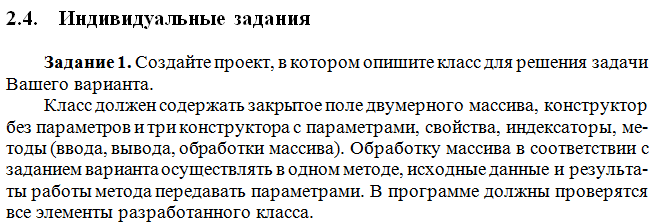
Выполнил студент группы АСОИ-181

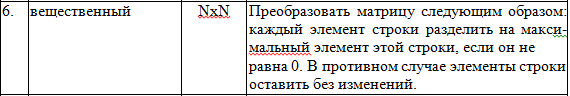
Самусев Д.А.

Проверил \_\_\_\_\_\_ Горбатенко Н.Н.

Могилёв 2020г

Цель работы – усвоение основных приемов создания классов с полями типа - массив; формированию знаний о возможности использования индексаторов для доступа к элементам закрытых массивов.





Код программы :

using System;  
  
namespace Lab2  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 var matrix = new Matrix();  
  
 string matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine(matrixAsString);  
  
 matrix.ChangeMatrix();  
 matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine($"Changed matrix: \n{matrixAsString}");  
  
 matrix = new Matrix(4);  
 matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine(matrixAsString);  
  
 matrix.ChangeMatrix();  
 matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine($"Changed matrix: \n{matrixAsString}");  
  
 matrix = new Matrix(4, 5);  
 matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine(matrixAsString);  
  
 matrix.ChangeMatrix();  
 matrixAsString = matrix.GetMatrixAsString();  
 Console.WriteLine($"Changed matrix: \n{matrixAsString}");  
  
 Console.WriteLine($"Element 1-1: {matrix[1, 1]}");  
  
 Console.ReadKey();  
 }  
 }  
}

namespace Lab2  
{  
 public class Matrix  
 {  
 private double[,] \_matrix;  
  
 public double[,] MatrixProperty  
 {  
 get => \_matrix;  
 set => \_matrix = value;  
 }  
  
 public double this[int row, int column]  
 {  
 get => \_matrix[row, column];  
 set => \_matrix[row, column] = value;  
 }  
  
 public Matrix()  
 {  
 \_matrix = new double[3, 3];  
 InitializeMatrix();  
 }  
  
 public Matrix(int side)  
 {  
 \_matrix = new double[side, side];  
 InitializeMatrix();  
 }  
  
 public Matrix(int height, int width)  
 {  
 \_matrix = new double[height, width];  
 InitializeMatrix();  
 }  
  
 public Matrix(double[,] matrix)  
 {  
 \_matrix = matrix;  
 }  
  
 public void ChangeMatrix()  
 {  
 for (var row = 0; row < \_matrix.GetLength(0); row++)  
 {  
 double max = FindMaxInRow(row);  
  
 if (max != 0)  
 {  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 \_matrix[row, column] /= max;  
 }  
 }  
 }  
 }  
  
 public string GetMatrixAsString()  
 {  
 string result = string.Empty;  
  
 for (var row = 0; row < \_matrix.GetLength(0); row++)  
 {  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 result += $"{\_matrix[row, column]} ";  
 }  
  
 result += "\n";  
 }  
  
 return result;  
 }  
  
 private void InitializeMatrix()  
 {  
 double element = 1;  
  
 for (var row = 0; row < \_matrix.GetLength(0); row++)  
 {  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 \_matrix[row, column] = element;  
 element++;  
 }  
 }  
 }  
  
 private double FindMaxInRow(int row)  
 {  
 double max = \_matrix[row, 0];  
  
 for (var column = 0; column < \_matrix.GetLength(1); column++)  
 {  
 double currentElement = \_matrix[row, column];  
  
 if (currentElement > max)  
 {  
 max = currentElement;  
 }  
 }  
  
 return max;  
 }  
 }  
}

