Лабораторная работа №4:

Уровень 1:

№3:

using System;

using static System.Console;

namespace L1\_task03

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(4, 4);

PrintMatrix(matrix);

double diagonalSum = DiagonalSum(matrix);

WriteLine($"Сумма диагоналей = {diagonalSum}");

}

static double DiagonalSum(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

double diagonalSum = 0;

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

{

double firstDiagonal = matrix[i, i];

double secondDiagonal = matrix[i, sideLength - 1 - i];

diagonalSum += firstDiagonal + secondDiagonal;

}

return diagonalSum;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows}\*{columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№6:

using System;

using static System.Console;

namespace L1\_task06

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(4, 7);

PrintMatrix(matrix);

double[] minValueArray = GetMinValueArray(matrix);

WriteLine("Массив с минимальными значениями из каждой строки:");

PrintArray(minValueArray);

}

static double[] GetMinValueArray(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

double[] minValueArray = new double[rows];

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

{

double minValue = matrix[i, 0];

for (int j = 0; j < columns; j++)

{

if (matrix[i, j] < minValue)

{

minValue = matrix[i, j];

}

}

minValueArray[i] = minValue;

}

return minValueArray;

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

using System;

using static System.Console;

namespace L1\_task06

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(4, 7);

PrintMatrix(matrix);

double[] minValueArray = GetMinValueArray(matrix);

WriteLine("Массив с минимальными значениями из каждой строки:");

PrintArray(minValueArray);

}

static double[] GetMinValueArray(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

double[] minValueArray = new double[rows];

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

{

double minValue = matrix[i, 0];

for (int j = 0; j < columns; j++)

{

if (matrix[i, j] < minValue)

{

minValue = matrix[i, j];

}

}

minValueArray[i] = minValue;

}

return minValueArray;

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№12:

using System;

using static System.Console;

namespace L1\_task12

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(6, 7);

PrintMatrix(matrix);

double[,] erasedMatrx = EraseMaxValueRowAndColumn(matrix);

WriteLine("Матрица без строки и столбца с пересечением на максимальном элемента:");

PrintMatrix(erasedMatrx);

}

static double[,] EraseMaxValueRowAndColumn(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

int maxValueRow = 0;

int maxValueColumn = 0;

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

{

for (int j = 0; j < columns; j++)

{

if (matrix[i, j] > matrix[maxValueRow, maxValueColumn])

{

maxValueRow = i;

maxValueColumn = j;

}

}

}

WriteLine($"Максимальное значение пересекается в [{maxValueRow}, {maxValueColumn}]");

WriteLine($"Или же в {maxValueRow + 1}-й строке и в {maxValueColumn + 1}-й столбце");

double[,] erasedMatrix = new double[rows - 1, columns - 1];

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

{

for (int j = 0; j < maxValueColumn; j++)

{

erasedMatrix[i, j] = matrix[i, j];

}

}

for (int i = maxValueRow + 1; i < rows; i++)

{

for (int j = maxValueColumn + 1; j < columns; j++)

{

erasedMatrix[i - 1, j - 1] = matrix[i, j];

}

}

return erasedMatrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№13:

using System;

using static System.Console;

namespace L1\_task13

{

internal class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(5, 5);

PrintMatrix(matrix);

int columnMaxValue = GetColumnWithMaxDiagonalValue(matrix);

WriteLine($"Столбец с максимальным элемент на диагонали = [{columnMaxValue}]");

WriteLine($"Он же {columnMaxValue + 1}-й столбец");

double[,] swappedMatrix = SwapColumns(matrix, 3, columnMaxValue);

PrintMatrix(swappedMatrix);

}

static int GetColumnWithMaxDiagonalValue(double[,] matrix)

{

int maxValueColumn = 0;

int sideLength = matrix.GetLength(0);

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

{

if (matrix[i, i] > matrix[maxValueColumn, maxValueColumn])

{

maxValueColumn = i;

}

if (matrix[i, sideLength - 1 - i] > matrix[maxValueColumn, maxValueColumn])

{

maxValueColumn = i;

}

}

return maxValueColumn;

}

static double[,] SwapColumns(double[,] matrix, int a, int b)

{

int rows = matrix.GetLength(0);

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

{

double buffer = matrix[i, a];

matrix[i, a] = matrix[i, b];

matrix[i, b] = buffer;

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№17:

using System;

using static System.Console;

namespace L1\_task17

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrixUserCount();

PrintMatrix(matrix);

double[,] minValueFirstColumnMatrix = SetMinValuesToFirstColumn(matrix);

WriteLine("Матрица со строками, где первый элемент = мин. значению:");

PrintMatrix(minValueFirstColumnMatrix);

}

static double[,] SetMinValuesToFirstColumn(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

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

{

int minValueIndex = 0;

for (int j = 0; j < columns; j++)

{

if (matrix[i, j] < matrix[i, minValueIndex])

{

minValueIndex = j;

}

}

double minValue = matrix[i, minValueIndex];

for (int j = minValueIndex; j > 0; j--)

{

matrix[i, j] = matrix[i, j - 1];

}

matrix[i, 0] = minValue;

}

return matrix;

}

static double[,] RandomMatrixUserCount()

{

Write("Введите кол-во строк: ");

int rows = int.Parse(ReadLine());

Write("Введите кол-во столбцов: ");

int columns = int.Parse(ReadLine());

return RandomMatrix(rows, columns);

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№29:

using System;

using static System.Console;

using static System.Math;

namespace L1\_task29

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(5, 7);

PrintMatrix(matrix);

double[,] withoutColumnMatrix = DeleteColumnAfterMinValueRow(matrix, 1);

WriteLine("Матрица без столбца, после столбца с минимальным по модулю элементом во 2-й строке:");

PrintMatrix(withoutColumnMatrix);

}

static double[,] DeleteColumnAfterMinValueRow(double[,] matrix, int rowIndex)

{

int minValueColumn = 0;

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

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

{

if (Abs(matrix[rowIndex, i]) < Abs(matrix[rowIndex, minValueColumn]))

{

minValueColumn = i;

}

}

if (minValueColumn == columns - 1)

{

Write("Столбец с минимальным значением является последним, удаление столбца невозможно");

return matrix;

}

int deleteColumn = minValueColumn + 1;

double[,] newMatrix = new double[rows, columns - 1];

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

{

for (int j = 0; j < deleteColumn; j++)

{

newMatrix[i, j] = matrix[i, j];

}

for (int j = deleteColumn + 1; j < columns; j++)

{

newMatrix[i, j - 1] = matrix[i, j];

}

}

return newMatrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№31:

using System;

using static System.Console;

using static System.Math;

namespace L1\_task31

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrixWithoutLastColumn(5, 8);

PrintMatrix(matrix);

double[] insertionVector = new double[5] { 999, 999, 999, 999, 999 };

double[,] filledMatrix = AddVectorAfterColumnMaxValue(matrix, insertionVector, 4);

WriteLine("Матрица, дозаполненная вектором после столбца с минимальным элементом в 5-й строке:");

PrintMatrix(filledMatrix);

}

static double[,] AddVectorAfterColumnMaxValue(double[,] matrix, double[] vector, int rowIndex)

{

int minValueColumn = 0;

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

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

{

if (matrix[rowIndex, i] < matrix[rowIndex, minValueColumn])

{

minValueColumn = i;

}

}

for (int j = columns - 1; j > minValueColumn + 1; j--)

{

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

{

matrix[i, j] = matrix[i, j - 1];

}

}

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

{

matrix[i, minValueColumn + 1] = vector[i];

}

return matrix;

}

static double[,] RandomMatrixWithoutLastColumn(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns - 1; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

Уровень 2:

№7:

using System;

using static System.Console;

namespace L2\_task07

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(6, 6);

PrintMatrix(matrix);

double[,] zeroedMatrix = ZeroUpperDiagonalAndMaxValue(matrix);

WriteLine("Матрица, где элемента правее главной диагонали (и выже максимального значения диагонали) обнулены:");

PrintMatrix(zeroedMatrix);

}

static double[,] ZeroUpperDiagonalAndMaxValue(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

int maxValueIndex = 0;

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

{

if (matrix[i, i] > matrix[maxValueIndex, maxValueIndex])

{

maxValueIndex = i;

}

}

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

{

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

{

matrix[i, j] = 0;

}

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№8:

using System;

using static System.Console;

namespace L2\_task08

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(6, 6);

PrintMatrix(matrix);

double[,] swappedMatrix = SwapRowNeighbors(matrix);

WriteLine("Матрица с поменяннами местами соседними строками (1 и 2, 3 и 4, 5 и 6):");

PrintMatrix(swappedMatrix);

}

static double[,] SwapRowNeighbors(double[,] matrix)

{

int rows = matrix.GetLength(0);

int colunms = matrix.GetLength(1);

for (int i = 0; i < rows; i += 2)

{

for (int j = 0; j < colunms; j++)

{

double temp = matrix[i, j];

matrix[i, j] = matrix[i + 1, j];

matrix[i + 1, j] = temp;

}

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

№9:

using System;

using static System.Console;

namespace L2\_task09

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

double[,] matrix = RandomMatrix(6, 7);

PrintMatrix(matrix);

double[,] swapRowMatrix = SwapRows(matrix);

WriteLine("Матрица с перевёрнутыми строками:");

PrintMatrix(swapRowMatrix);

}

static double[,] SwapRows(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

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

{

for (int j = 0; j < columns / 2; j++)

{

double temp = matrix[i, j];

matrix[i, j] = matrix[i, columns - j - 1];

matrix[i, columns - j - 1] = temp;

}

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

}

}

Уровень 3:

№1:

using System;

using static System.Console;

namespace L3\_task01

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomMatrix(7, 5);

PrintMatrix(matrix);

double[,] sortedMatrix = SortRowsByMinValue(matrix);

WriteLine("Матрица, где строки сортированы по минимальному значению каждой строки:");

PrintMatrix(sortedMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomOneDimensionalMatrix(7, 5);

PrintOneDimensionalMatrix(odMatrix, 7, 5);

double[] sortedOdMatrix = SortRowsByMinValueOneDimensional(odMatrix, 7, 5);

WriteLine("Матрица, где строки сортированы по минимальному значению каждой строки:");

PrintOneDimensionalMatrix(sortedOdMatrix, 7, 5);

}

static double[,] SortRowsByMinValue(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

double[][] raggedMatrix = new double[rows][];

double[] minValues = new double[rows];

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

{

raggedMatrix[i] = new double[columns];

double minValue = matrix[i, 0];

for (int j = 0; j < columns; j++)

{

raggedMatrix[i][j] = matrix[i, j];

if (matrix[i, j] < minValue)

{

minValue = matrix[i, j];

}

}

minValues[i] = minValue;

}

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

{

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

{

if (minValues[j] < minValues[j + 1])

{

double temp = minValues[j + 1];

minValues[j + 1] = minValues[j];

minValues[j] = temp;

double[] tempArray = raggedMatrix[j + 1];

raggedMatrix[j + 1] = raggedMatrix[j];

raggedMatrix[j] = tempArray;

}

}

}

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = raggedMatrix[i][j];

}

}

return matrix;

}

static double[] SortRowsByMinValueOneDimensional(double[] matrix, int rows, int columns)

{

double[][] raggedMatrix = new double[rows][];

double[] minValues = new double[rows];

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

{

raggedMatrix[i] = new double[columns];

double minValue = matrix[columns \* i];

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

raggedMatrix[i][j] = matrix[index];

if (matrix[index] < minValue)

{

minValue = matrix[index];

}

}

minValues[i] = minValue;

}

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

{

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

{

if (minValues[j] < minValues[j + 1])

{

double temp = minValues[j + 1];

minValues[j + 1] = minValues[j];

minValues[j] = temp;

double[] tempArray = raggedMatrix[j + 1];

raggedMatrix[j + 1] = raggedMatrix[j];

raggedMatrix[j] = tempArray;

}

}

}

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

{

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

matrix[index] = raggedMatrix[i][j];

}

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

}

}

№2:

using System;

using static System.Console;

namespace L3\_task02

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomMatrix(6, 6);

PrintMatrix(matrix);

double[,] zeroedMatrix = FillBordersZeroes(matrix);

WriteLine("Матрица с 0-границей:");

PrintMatrix(zeroedMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomOneDimensionalMatrix(6, 6);

PrintOneDimensionalMatrix(odMatrix, 6, 6);

double[] zeroedOdMatrix = FillBorderZeroesOneDimensional(odMatrix, 6);

WriteLine("Матрица с 0-границей:");

PrintOneDimensionalMatrix(zeroedOdMatrix, 6, 6);

}

static double[,] FillBordersZeroes(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

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

{

matrix[i, 0] = 0;

matrix[0, i] = 0;

matrix[sideLength - 1, i] = 0;

matrix[i, sideLength - 1] = 0;

}

return matrix;

}

static double[] FillBorderZeroesOneDimensional(double[] matrix, int sideLength)

{

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

{

matrix[Index(i, 0, sideLength)] = 0;

matrix[Index(0, i, sideLength)] = 0;

matrix[Index(sideLength - 1, i, sideLength)] = 0;

matrix[Index(i, sideLength -1, sideLength)] = 0;

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

}

}

№3:

using System;

using static System.Console;

namespace L3\_task03

{

class Program

{

static int \_length;

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomUserMatrix();

PrintMatrix(matrix);

double[] sums = GetDiagonalSumms(matrix);

WriteLine("Суммы диагоналей:");

PrintArray(sums);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomUserOneDimensionalMatrix();

PrintOneDimensionalMatrix(odMatrix, \_length, \_length);

double[] odSums = GetDiagonalSummsOneDimensional(odMatrix);

WriteLine("Суммы диагоналей:");

PrintArray(odSums);

}

static double[] GetDiagonalSumms(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

double[] results = new double[2 \* sideLength - 1];

int index = -1;

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

{

index++;

double sum = 0;

for (int j = 0; j + i < sideLength; j++)

{

sum += matrix[j, j + i];

}

results[index] = sum;

}

for (int i = 1; i < sideLength; i++)

{

index++;

double sum = 0;

for (int j = 0; j + i < sideLength; j++)

{

sum += matrix[j + i, j];

}

results[index] = sum;

}

return results;

}

static double[] GetDiagonalSummsOneDimensional(double[] matrix)

{

int sideLength = \_length;

double[] results = new double[2 \* sideLength - 1];

int index = -1;

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

{

index++;

double sum = 0;

for (int j = 0; j + i < sideLength; j++)

{

sum += matrix[Index(j, j + i, sideLength)];

}

results[index] = sum;

}

for (int i = 1; i < sideLength; i++)

{

index++;

double sum = 0;

for (int j = 0; j + i < sideLength; j++)

{

sum += matrix[Index(j + i, j, sideLength)];

}

results[index] = sum;

}

return results;

}

static double[,] RandomUserMatrix()

{

Write("Введите кол-во строк и столбцов: ");

int sideLength = int.Parse(ReadLine());

return RandomMatrix(sideLength, sideLength);

}

static double[] RandomUserOneDimensionalMatrix()

{

Write("Введите кол-во строк и столбцов: ");

int sideLength = int.Parse(ReadLine());

\_length = sideLength;

return RandomOneDimensionalMatrix(sideLength, sideLength);

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = Index(i, j, columns);

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

}

}

№4:

using System;

using static System.Console;

namespace L3\_task04

{

class Program

{

static int \_length;

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomUserMatrix();

PrintMatrix(matrix);

double[,] filledMatrix = FillDownAreaOnes(matrix);

WriteLine("Матрица, заполненная единицами:");

PrintMatrix(filledMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomUserOneDimensionalMatrix();

PrintOneDimensionalMatrix(odMatrix, \_length, \_length);

double[] filledOdMatrix = FillDownAreaOnesOneDimensional(odMatrix);

WriteLine("Матрица, заполненная единицами:");

PrintOneDimensionalMatrix(filledOdMatrix, \_length, \_length);

}

static double[,] FillDownAreaOnes(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

for (int i = rows / 2; i < rows; i++)

{

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

{

matrix[i, j] = 1;

}

}

return matrix;

}

static double[] FillDownAreaOnesOneDimensional(double[] matrix)

{

int rows = \_length;

for (int i = rows / 2; i < rows; i++)

{

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

{

matrix[Index(i, j, rows)] = 1;

}

}

return matrix;

}

static double[,] RandomUserMatrix()

{

Write("Введите кол-во строк и столбцов: ");

int sideLength = int.Parse(ReadLine());

return RandomMatrix(sideLength, sideLength);

}

static double[] RandomUserOneDimensionalMatrix()

{

Write("Введите кол-во строк и столбцов: ");

int sideLength = int.Parse(ReadLine());

\_length = sideLength;

return RandomOneDimensionalMatrix(sideLength, sideLength);

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = Index(i, j, columns);

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

}

}

№8:

using System;

using static System.Console;

namespace L3\_task08

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomMatrix(7, 5);

PrintMatrix(matrix);

double[,] sortedMatrix = SortRowsByPositiveCount(matrix);

WriteLine("Матрица, где строки отсортированы по кол-ву положительных чисел по убыванию:");

PrintMatrix(sortedMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomOneDimensionalMatrix(7, 5);

PrintOneDimensionalMatrix(odMatrix, 7, 5);

double[] sortedOdMatrix = SortRowsByPositiveCountOneDimensional(odMatrix, 7, 5);

WriteLine("Матрица, где строки отсортированы по кол-ву положительных чисел по убыванию:");

PrintOneDimensionalMatrix(sortedOdMatrix, 7, 5);

}

static double[,] SortRowsByPositiveCount(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

double[][] raggedMatrix = new double[rows][];

double[] positiveCounts = new double[rows];

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

{

raggedMatrix[i] = new double[columns];

double positiveCount = 0;

for (int j = 0; j < columns; j++)

{

raggedMatrix[i][j] = matrix[i, j];

if (matrix[i, j] > 0)

{

positiveCount++;

}

}

positiveCounts[i] = positiveCount;

}

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

{

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

{

if (positiveCounts[j] < positiveCounts[j + 1])

{

double temp = positiveCounts[j + 1];

positiveCounts[j + 1] = positiveCounts[j];

positiveCounts[j] = temp;

double[] tempArray = raggedMatrix[j + 1];

raggedMatrix[j + 1] = raggedMatrix[j];

raggedMatrix[j] = tempArray;

}

}

}

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = raggedMatrix[i][j];

}

}

return matrix;

}

static double[] SortRowsByPositiveCountOneDimensional(double[] matrix, int rows, int columns)

{

double[][] raggedMatrix = new double[rows][];

double[] positiveCounts = new double[rows];

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

{

raggedMatrix[i] = new double[columns];

double positiveCount = 0;

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

raggedMatrix[i][j] = matrix[index];

if (matrix[index] > 0)

{

positiveCount++;

}

}

positiveCounts[i] = positiveCount;

}

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

{

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

{

if (positiveCounts[j] < positiveCounts[j + 1])

{

double temp = positiveCounts[j + 1];

positiveCounts[j + 1] = positiveCounts[j];

positiveCounts[j] = temp;

double[] tempArray = raggedMatrix[j + 1];

raggedMatrix[j + 1] = raggedMatrix[j];

raggedMatrix[j] = tempArray;

}

}

}

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

{

for (int j = 0; j < columns; j++)

{

int index = columns \* i + j;

matrix[index] = raggedMatrix[i][j];

}

}

return matrix;

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = Index(i, j, columns);

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

}

}

№10:

using System;

using static System.Console;

namespace L3\_task10

{

class Program

{

static int \_rows, \_columns;

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomUserMatrix();

PrintMatrix(matrix);

double[,] sortedMatrix = SortRowsEvenOdd(matrix);

WriteLine("Матрица, где чётные индексы в строках отсортированы по убыванию, нечётные - по возрастанию:");

PrintMatrix(sortedMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomUserOneDimensionalMatrix();

PrintOneDimensionalMatrix(odMatrix, \_rows, \_columns);

double[] sortedOdMatrix = SortRowsEvenOddOneDimensional(odMatrix, \_rows, \_columns);

WriteLine("Матрица, где чётные индексы в строках отсортированы по убыванию, нечётные - по возрастанию:");

PrintOneDimensionalMatrix(sortedOdMatrix, \_rows, \_columns);

}

static double[,] SortRowsEvenOdd(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

for (int k = 0; k < rows; k++)

{

for (int i = 0; i < columns; i += 2)

{

for (int j = 0; j < columns - 2; j += 2)

{

if ((matrix[k, j] < matrix[k, j + 2]))

{

double temp = matrix[k, j + 2];

matrix[k, j + 2] = matrix[k, j];

matrix[k, j] = temp;

}

}

for (int j = 0; j + 1 < columns - 2; j += 2)

{

j += 1;

if ((matrix[k, j] > matrix[k, j + 2]))

{

double temp = matrix[k, j + 2];

matrix[k, j + 2] = matrix[k, j];

matrix[k, j] = temp;

}

j -= 1;

}

}

}

return matrix;

}

static double[] SortRowsEvenOddOneDimensional(double[] matrix, int rows, int columns)

{

for (int k = 0; k < rows; k++)

{

for (int i = 0; i < columns; i += 2)

{

for (int j = 0; j < columns - 2; j += 2)

{

if ((matrix[Index(k, j, columns)] < matrix[Index(k, j + 2, columns)]))

{

double temp = matrix[Index(k, j + 2, columns)];

matrix[Index(k, j + 2, columns)] = matrix[Index(k, j, columns)];

matrix[Index(k, j, columns)] = temp;

}

}

for (int j = 0; j + 1 < columns - 2; j += 2)

{

j += 1;

if ((columns - j >= 2) && (matrix[Index(k, j, columns)] > matrix[Index(k, j + 2, columns)]))

{

double temp = matrix[Index(k, j + 2, columns)];

matrix[Index(k, j + 2, columns)] = matrix[Index(k, j, columns)];

matrix[Index(k, j, columns)] = temp;

}

j -= 1;

}

}

}

return matrix;

}

static double[,] RandomUserMatrix()

{

Write("Введите кол-во строк: ");

int rows = int.Parse(ReadLine());

Write("Введите кол-во столбцов: ");

int columns = int.Parse(ReadLine());

return RandomMatrix(rows, columns);

}

static double[] RandomUserOneDimensionalMatrix()

{

Write("Введите кол-во строк: ");

int rows = int.Parse(ReadLine());

\_rows = rows;

Write("Введите кол-во столбцов: ");

int columns = int.Parse(ReadLine());

\_columns = columns;

return RandomOneDimensionalMatrix(rows, columns);

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

matrix[i, j] = \_random.Next(MinValue, MaxValue);

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

matrix[i] = \_random.Next(MinValue, MaxValue);

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = Index(i, j, columns);

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

}

}

№11:

using System;

using static System.Console;

namespace L3\_task11

{

class Program

{

static int \_rows, \_columns;

static Random \_random = new Random();

const int MinValue = -5;

const int MaxValue = 30;

static void Main(string[] args)

{

WriteLine("A) В виде двумерного массива:\n");

double[,] matrix = RandomUserMatrix();

PrintMatrix(matrix);

double[,] sortedMatrix = RowsWithoutZeroes(matrix);

WriteLine("Матрица, без строк с нулевыми элементами:");

PrintMatrix(sortedMatrix);

WriteLine("\nБ) В виде одномерной последовательности:\n");

double[] odMatrix = RandomUserOneDimensionalMatrix();

PrintOneDimensionalMatrix(odMatrix, \_rows, \_columns);

double[] sortedOdMatrix = RowsWithoutZeroesOneDimensional(odMatrix, \_rows, \_columns);

WriteLine("Матрица, без строк с нулевыми элементами:");

PrintOneDimensionalMatrix(sortedOdMatrix, \_rows, \_columns);

}

static double[,] RowsWithoutZeroes(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

bool[] hasZeroes = new bool[rows];

int withZeroCount = 0;

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

{

bool hasZero = false;

for (int j = 0; j < columns; j++)

{

if (matrix[i, j] == 0)

{

withZeroCount++;

hasZero = true;

break;

}

}

hasZeroes[i] = hasZero;

}

double[,] withoutZeroMatrix = new double[rows - withZeroCount, columns];

int index = -1;

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

{

if (!hasZeroes[i])

{

index++;

for (int j = 0; j < columns; j++)

{

withoutZeroMatrix[index, j] = matrix[i, j];

}

}

}

return withoutZeroMatrix;

}

static double[] RowsWithoutZeroesOneDimensional(double[] matrix, int rows, int columns)

{

bool[] hasZeroes = new bool[rows];

int withZeroCount = 0;

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

{

bool hasZero = false;

for (int j = 0; j < columns; j++)

{

if (matrix[Index(i, j, columns)] == 0)

{

withZeroCount++;

hasZero = true;

break;

}

}

hasZeroes[i] = hasZero;

}

double[] withoutZeroMatrix = new double[(rows - withZeroCount) \* columns];

\_rows -= withZeroCount;

int index = -1;

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

{

if (!hasZeroes[i])

{

index++;

for (int j = 0; j < columns; j++)

{

withoutZeroMatrix[Index(index, j, columns)] = matrix[Index(i, j, columns)];

}

}

}

return withoutZeroMatrix;

}

static double[,] RandomUserMatrix()

{

Write("Введите кол-во строк: ");

int rows = int.Parse(ReadLine());

Write("Введите кол-во столбцов: ");

int columns = int.Parse(ReadLine());

return RandomMatrix(rows, columns);

}

static double[] RandomUserOneDimensionalMatrix()

{

Write("Введите кол-во строк: ");

int rows = int.Parse(ReadLine());

\_rows = rows;

Write("Введите кол-во столбцов: ");

int columns = int.Parse(ReadLine());

\_columns = columns;

return RandomOneDimensionalMatrix(rows, columns);

}

static double[,] RandomMatrix(int rows, int columns)

{

double[,] matrix = new double[rows, columns];

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

{

for (int j = 0; j < columns; j++)

{

int value = \_random.Next(MinValue, MaxValue);

matrix[i, j] = value > 0? value : 0;

}

}

return matrix;

}

static double[] RandomOneDimensionalMatrix(int rows, int columns)

{

double[] matrix = new double[rows \* columns];

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

{

int value = \_random.Next(MinValue, MaxValue);

matrix[i] = value > 0 ? value : 0;

}

return matrix;

}

static int Index(int i, int j, int columns)

{

return columns \* i + j;

}

static void PrintMatrix(double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

Write($"{matrix[i, j]},\t");

}

Write("\n");

}

}

static void PrintOneDimensionalMatrix(double[] matrix, int rows, int columns)

{

WriteLine($"Матрица {rows} на {columns}:");

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

{

for (int j = 0; j < columns; j++)

{

int index = Index(i, j, columns);

Write($"{matrix[index]},\t");

}

Write("\n");

}

}

static void PrintArray(double[] array)

{

if (array.Length == 0)

{

WriteLine("\* Массив пустой \*");

}

else

{

WriteLine($"Массив из {array.Length} элементов:");

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

{

WriteLine($"[{i}] = {array[i]}");

}

}

}

}

}