Лабораторная работа №5:

Уровень 1:

№1:

using System;

using static System.Console;

namespace L1\_task01

{

class Program

{

static void Main(string[] args)

{

int teamCount = 5;

WriteLine("Отобразить команду из 5-ти человек можно следующее кол-во раз:");

WriteLine($"Из 8-ти кандидатов: {CombinationCount(teamCount, 8)}");

WriteLine($"Из 10-ти кандидатов: {CombinationCount(teamCount, 10)}");

WriteLine($"Из 11-ти кандидатов: {CombinationCount(teamCount, 11)}");

}

static int CombinationCount(int k, int n)

{

return Factorial(n) / (Factorial(k) \* Factorial(n - k));

}

static int Factorial(int n)

{

int factorial = 1;

for (int i = 2; i <= n; i++)

factorial \*= i;

return factorial;

}

}

}

№2:

using System;

using static System.Console;

using static System.Math;

namespace L1\_task02

{

class Program

{

static int \_orderNumber = 0;

static void Main(string[] args)

{

double firstTriangleArea = TriangleAreaUserInput();

double secondTriangleArea = TriangleAreaUserInput();

if (firstTriangleArea > secondTriangleArea)

{

WriteLine($"\nПлощадь 1-го треугольника максимальна = {firstTriangleArea}");

}

else if (secondTriangleArea > firstTriangleArea)

{

WriteLine($"\nПлощадь 2-го треугольника максимальна = {secondTriangleArea}");

}

else

{

WriteLine("\nПлощади обоих треугольников равны");

}

}

static double TriangleAreaUserInput()

{

\_orderNumber++;

WriteLine($"{\_orderNumber}-й треугольник:\n");

Write("Введите длину A: ");

double a = double.Parse(ReadLine());

Write("Введите длину A: ");

double b = double.Parse(ReadLine());

Write("Введите длину A: ");

double c = double.Parse(ReadLine());

double area = GetTriangleArea(a, b, c);

WriteLine($"Площадь треугольника = {area}");

return area;

}

static double GetTriangleArea(double a, double b, double c)

{

double p = (a + b + c) / 2;

return Sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

}

}

Уровень 2:

№6:

using System;

using static System.Console;

namespace L2\_task06

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

var a = RandomArray(7);

PrintArray(a, "Массив А:");

var b = RandomArray(8);

PrintArray(b, "Массив B:");

MergeArrays(ref a, ref b);

PrintArray(a, "Объеденённые массивы в A:");

}

static void MergeArrays(ref double[] destination, ref double[] source)

{

RemoveMaxValue(ref destination);

RemoveMaxValue(ref source);

var mergedArray = new double[destination.Length + source.Length];

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

{

if (i >= destination.Length)

{

mergedArray[i] = source[i - destination.Length];

}

else

{

mergedArray[i] = destination[i];

}

}

destination = mergedArray;

source = new double[] { };

}

static void RemoveMaxValue(ref double[] array)

{

RemoveAt(ref array, MaxPosition(array));

}

static void RemoveAt(ref double[] array, int position)

{

var smallerArray = new double[array.Length - 1];

int offset = 0;

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

{

if (i == position)

{

offset++;

continue;

}

smallerArray[i - offset] = array[i];

}

array = smallerArray;

}

static int MaxPosition(double[] array)

{

int maxValuePosition = 0;

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

{

if (array[i] > array[maxValuePosition])

{

maxValuePosition = i;

}

}

return maxValuePosition;

}

static double[] RandomArray(int itemCount)

{

var array = new double[itemCount];

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

{

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

}

return array;

}

static void PrintArray(double[] array, string startMessage = "")

{

if (startMessage != "")

{

WriteLine(startMessage);

}

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 L2\_task10

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

var matrix = RandomUserMatrix();

PrintMatrix(matrix, "Сформированная матрица:");

RemoveMinMaxColumns(ref matrix);

PrintMatrix(matrix, "Матрица без 2-х столбцов:");

}

static void RemoveMinMaxColumns(ref double[,] matrix)

{

int maxValueColumn = ColumnMaxValueUnderDiagonal(matrix);

int minValueColumn = ColumnMinValueAboveDiagonal(matrix);

RemoveColumnAt(ref matrix, maxValueColumn);

if (maxValueColumn > minValueColumn)

{

RemoveColumnAt(ref matrix, minValueColumn);

}

else if (maxValueColumn < minValueColumn)

{

RemoveColumnAt(ref matrix, minValueColumn - 1);

}

}

static int ColumnMaxValueUnderDiagonal(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

double maxValue = matrix[1, 0];

int maxValueColumn = 1;

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

{

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

{

if (matrix[j, j + i] > maxValue)

{

maxValue = matrix[j + i, j];

maxValueColumn = j;

}

}

}

WriteLine($"Столбец с максимальным значением под диагональю = {maxValueColumn + 1}");

return maxValueColumn;

}

static int ColumnMinValueAboveDiagonal(double[,] matrix)

{

int sideLength = matrix.GetLength(0);

double minValue = matrix[0, 1];

int minValueColumn = 1;

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

{

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

{

if (matrix[j, j + i] < minValue)

{

minValue = matrix[j, j + i];

minValueColumn = j + i;

}

}

}

WriteLine($"Столбец с минимальным значением над диагональю = {minValueColumn + 1}");

return minValueColumn;

}

static void RemoveColumnAt(ref double[,] matrix, int columnPosition)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

var smallerMatrix = new double[rows, columns - 1];

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

{

int offset = 0;

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

{

if (j == columnPosition)

{

offset++;

continue;

}

smallerMatrix[i, j - offset] = matrix[i, j];

}

}

matrix = smallerMatrix;

}

static double[,] RandomUserMatrix()

{

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

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

return RandomMatrix(rows, rows);

}

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;

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix, string message = "")

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

if (message != "")

{

WriteLine(message);

}

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");

}

}

}

}

№23:

using System;

using static System.Console;

namespace L2\_task23

{

class Program

{

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

var matrix = RandomUserMatrix();

PrintMatrix(matrix, "Сформированная матрица:");

MultipleMaxAndDivideAll(ref matrix);

PrintMatrix(matrix, "Матрица, где 5 наибольших элементов умножено на 2, а остальные - поделены на 2");

}

static void MultipleMaxAndDivideAll(ref double[,] matrix)

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

int[,] maxValuePositions = new int[5, 2];

for (int k = 0; k < maxValuePositions.GetLength(0); k++)

{

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

{

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

{

if (k == 0)

{

if (matrix[i, j] > matrix[maxValuePositions[k, 0], maxValuePositions[k, 1]])

{

maxValuePositions[k, 0] = i;

maxValuePositions[k, 1] = j;

continue;

}

}

else if ((matrix[i, j] > matrix[maxValuePositions[k, 0], maxValuePositions[k, 1]]) &&

(matrix[i, j] < matrix[maxValuePositions[k - 1, 0], maxValuePositions[k - 1, 1]]))

{

maxValuePositions[k, 0] = i;

maxValuePositions[k, 1] = j;

continue;

}

}

}

}

WriteLine("5 максимальных элементов: ");

for (int i = 0; i < maxValuePositions.GetLength(0); i++)

{

WriteLine($"[{i}] = {matrix[maxValuePositions[i, 0], maxValuePositions[i, 1]]}");

}

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

{

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

{

bool isOneOfMax = false;

for (int k = 0; k < maxValuePositions.GetLength(0); k++)

{

if ((i == maxValuePositions[k, 0]) && (j == maxValuePositions[k, 1]))

{

isOneOfMax = true;

break;

}

}

if (isOneOfMax)

{

matrix[i, j] \*= 2;

}

else

{

matrix[i, j] /= 2;

}

}

}

}

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[,] 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;

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix, string message = "")

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

if (message != "")

{

WriteLine(message);

}

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;

using static System.Math;

namespace L3\_task01

{

class Program

{

delegate double Function(double x);

static void Main(string[] args)

{

const double A = 0.1;

const double B = 1;

const double deltaX = 0.1;

for (double x = A; x <= B; x += deltaX)

{

double y = FunctionValue(StandardFunction, x);

double s = FunctionValue(TaylorRowFunction, x);

WriteLine($"x = {x}: y = {y}; s = {s}");

}

}

static double FunctionValue(Function function, double argument)

{

return function(argument);

}

static double StandardFunction(double x)

{

return Exp(Cos(x)) \* Cos(Sin(x));

}

static double TaylorRowFunction(double x)

{

const double epsilon = 0.0001;

double sum = 1;

double rowItemValue = 1;

int i = 0;

int denomenator = 1;

do

{

i++;

denomenator \*= i;

rowItemValue = Cos(i \* x) / denomenator;

sum += rowItemValue;

}

while (Abs(rowItemValue) > epsilon);

return sum;

}

}

}

№2:

using System;

using static System.Console;

namespace L3\_task02

{

class Program

{

delegate bool Comparer(double a, double b);

delegate void SortMatrix(ref double[,] matrix, int rowIndex, Comparer comparer);

static Random \_random = new Random();

const int MinValue = -30;

const int MaxValue = 30;

static void Main(string[] args)

{

var matrix = RandomUserMatrix();

PrintMatrix(matrix, "Исходная матрица:");

SortByIsRowEven(ref matrix);

PrintMatrix(matrix, "Упорядоченная матрица: ");

}

static bool IsMoreComparer(double a, double b)

{

return a > b;

}

static bool IsLessComparer(double a, double b)

{

return a < b;

}

static void SortMatrixRow(ref double[,] matrix, int rowIndex, Comparer comparer)

{

int columns = matrix.GetLength(1);

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

{

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

{

if (comparer(matrix[rowIndex, j], matrix[rowIndex, j + 1]))

{

double temp = matrix[rowIndex, j + 1];

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

matrix[rowIndex, j] = temp;

}

}

}

}

static void SortByIsRowEven(ref double[,] matrix)

{

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

{

if ((i + 1) % 2 == 0)

{

SortMatrixRow(ref matrix, i, IsMoreComparer);

}

else

{

SortMatrixRow(ref matrix, i, IsLessComparer);

}

}

}

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[,] 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;

}

}

return matrix;

}

static void PrintMatrix(double[,] matrix, string message = "")

{

int rows = matrix.GetLength(0);

int columns = matrix.GetLength(1);

if (message != "")

{

WriteLine(message);

}

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");

}

}

}

}