

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра программных систем

**Задание №4**

**Метод Гаусса**

Выполнил: Мороз

Илья Олегович

Группа № K3220

Преподаватель: Иванов Сергей Евгеньевич

Санкт-Петербург

2020

**Задание:** реализовать решение системы уравнений с помощью метода Гаусса.

**Исходный код:**

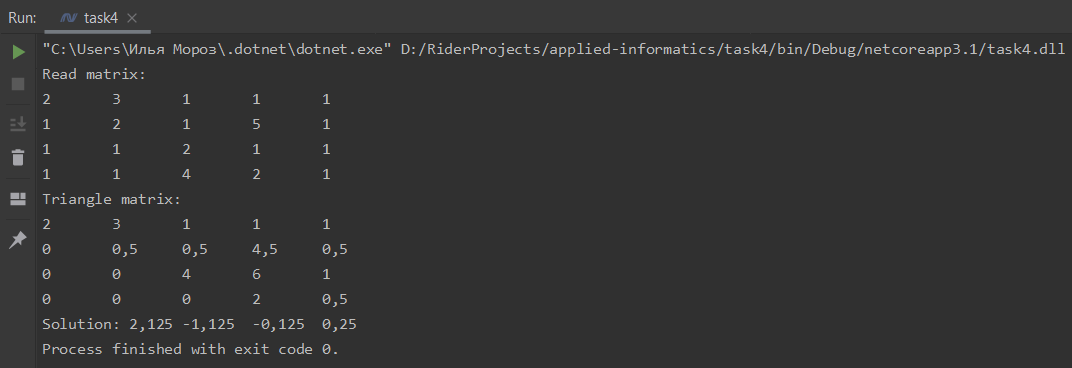
**Program.cs**

using System;  
using System.IO;  
using System.Linq;  
  
namespace task4  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 double[][] matrix = ReadMatrixFromFile("input.txt");  
 Console.WriteLine("Read matrix:");  
 PrintMatrix(matrix);  
 double[] resolve = Solve(matrix);  
 PrintSolution(resolve);  
 }  
  
 */\*  
 \* Read matrix from file  
 \* First line - count of variables in matrix  
 \*/* private static double[][] ReadMatrixFromFile(string filename)  
 {  
 try  
 {  
 StreamReader inputFile = new StreamReader(filename);  
  
 */\*  
 \* Get count of variables from file  
 \*/* int variableCount = Int32.Parse(inputFile.ReadLine());  
 double[][] matrix = new double[variableCount][];  
  
 */\*  
 \* Read lines with information of incidences  
 \*/* for (int i = 0; i < variableCount; i++)  
 {  
 string inputLine = inputFile.ReadLine();  
  
 */\*  
 \* Throw exception if line is null  
 \*/* string[] splittedInputLine = (inputLine != null) ? inputLine.Split(" ") : throw new IOException();  
 if (splittedInputLine.Length != variableCount + 1) throw new IOException();  
 matrix[i] = splittedInputLine.Select(vortexInfo => Double.Parse(vortexInfo)).ToArray();  
 }  
  
 inputFile.Close();  
 return matrix;  
 }  
 catch (Exception e)  
 {  
 ErrorHandler.WriteErrorInConsole(e, e.Message);  
 }  
  
 return new double[0][];  
 }  
  
 */\*  
 \* Write matrix in console  
 \*/* private static void PrintMatrix(double[][] matrix)  
 {  
 for (int i = 0; i < matrix.Length; i++)  
 {  
 for (int j = 0; j < matrix[0].Length; j++)  
 {  
 Console.Write($"{matrix[i][j]}\t");  
 }  
 Console.WriteLine();  
 }  
 }  
  
 */\*  
 \* Write solution in console  
 \*/* private static void PrintSolution(double[] resolve)  
 {  
 if (resolve == null)  
 {  
 Console.WriteLine("The only solution to the system does not exist");  
 return;  
 }  
 Console.Write("Solution: ");  
 for (int i = 0; i < resolve.Length; i++)  
 {  
 Console.Write($"{resolve[i]}\t");  
 }  
 }  
  
 */\*  
 \* Subtract row #n  
 \* Return new matrix  
 \*/* private static void SubtractRow(ref double[][] matrix, int n)  
 {  
 double m = matrix[n][n];  
 for (int i = n + 1; i < matrix.Length; i++)  
 {  
 double t = matrix[i][n] / m;  
 for (int j = n; j < matrix[0].Length; j++)  
 {  
 matrix[i][j] = matrix[i][j] - matrix[n][j] \* t;  
 if (matrix[i][j] < 0.0001 && matrix[i][j] > 0.0001) matrix[i][j] = 0;  
 }  
 }  
 }  
  
 */\*  
 \* Get Lead in column #n  
 \*/* static void SelectLeading(ref double[][] matrix, int n)  
 {  
 int indexOfRowWithMax = n;  
 for (int i = n + 1; i < matrix.Length; i++)  
 {  
 if (Math.Abs(matrix[indexOfRowWithMax][n]) < Math.Abs(matrix[i][n]))  
 {  
 indexOfRowWithMax = i;  
 }  
 }  
  
 if (indexOfRowWithMax != n)  
 {  
 for (int i = n; i < matrix[0].Length; i++)  
 {  
 matrix[n][i] += matrix[indexOfRowWithMax][i] - (matrix[indexOfRowWithMax][i] = matrix[n][i]);  
 }  
 }  
 }  
  
 */\*  
 \* Get matrix in triangular view  
 \* Return false if can't find solution  
 \*/* private static bool TriangleMatrix(ref double[][] matrix)  
 {  
 for (int i = 1; i < matrix.Length; i++)  
 {  
 SelectLeading(ref matrix, i - 1);  
 if (Math.Abs(matrix[i - 1][i - 1]) > 0.0001)  
 {  
 SubtractRow(ref matrix, i - 1);  
 }  
 else  
 {  
 return false;  
 }  
 }  
  
 return true;  
 }  
  
 */\*  
 \* Return solution for matrix by Gauss method  
 \*/* private static double[] Solve(double[][] matrix)  
 {  
 if (!TriangleMatrix(ref matrix)) return null;  
 Console.WriteLine("Triangle matrix:");  
 PrintMatrix(matrix);  
 double[] resolve = new double[matrix.Length];  
 int countOfVariables = matrix[0].Length - 1;  
 for (int i = resolve.Length - 1; i >= 0; i--)  
 {  
 double sum = 0;  
 for (int j = i + 1; j < countOfVariables; j++)  
 {  
 sum += resolve[j] \* matrix[i][j];  
 }  
  
 resolve[i] = (matrix[i][countOfVariables] - sum) / matrix[i][i];  
 }  
  
 return resolve;  
 }  
 }  
}

**ErrorHandler.cs**

using System;  
  
namespace task2  
{  
 public class ErrorHandler  
 {  
 public static void WriteErrorInConsole(Exception exception, string message = "Something went wrong!")  
 {  
 Console.ForegroundColor = ConsoleColor.**Red**;  
 Console.WriteLine(message);  
 Console.WriteLine("Stack trace:");  
 Console.WriteLine(exception.StackTrace);  
 Console.ResetColor();  
 System.Environment.Exit(1);  
 }  
 }  
}

**Результат работы:**

****

**Вывод:** был реализовано нахождение решения системы методом Гаусса.

Исходный код также доступен на GitHub: [ilyamore88/applied-informatics/task4](https://github.com/ilyamore88/applied-informatics/tree/master/task4).