**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра математического обеспечения и применения**

Отчет по лабораторной работе №6

по курсу «Объектно-ориентированное программирование»

**«Создание графического интерфейса»**

Выполнил:

Студент гр.КТбо2-7

Серченко О. С.

Проверил:

Преподаватель по ОСиСПО

Тарасов С. А.

Таганрог 2020

Задание

Определить класс вещественных матриц (Matrix) с методами, реализующими сложение и умножение матриц, транспонирование. Также класс квадратичных матриц SquareMatrux с возможностью нахождения детерминанта.

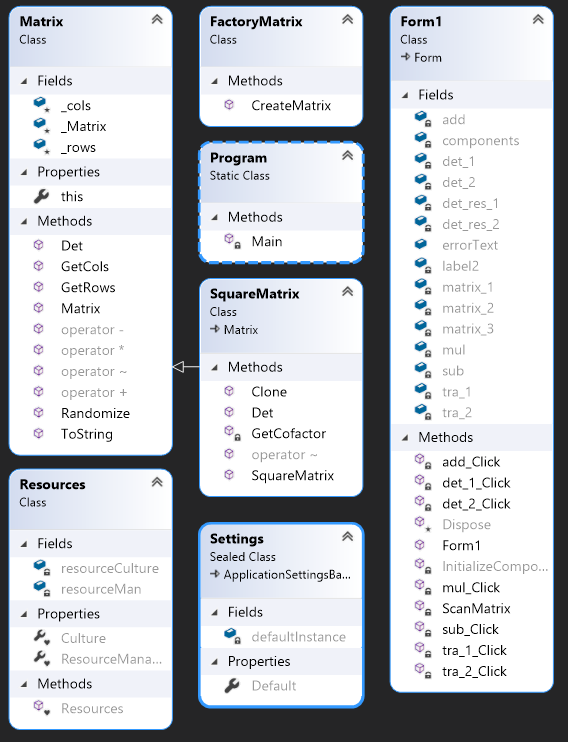
Спецификация классов

Спецификация классов представлена на диаграмме классов.

Зависимости и алгоритмы

Использованы стандартные алгоритмы.

Диаграмма классов



Листинг

Matrices.cs:

using System;

namespace Matrices

{

class Matrix

{

public uint GetRows()

{

return \_rows;

}

public uint GetCols()

{

return \_cols;

}

public void Randomize()

{

Random random = new Random();

for (uint i = 0; i < \_rows; i++)

{

for (uint j = 0; j < \_cols; j++)

{

\_Matrix[i][j] = random.NextDouble();

}

}

}

public virtual double Det()

{

throw new NotSupportedException();

}

public double[] this[uint i]

{

get { return \_Matrix[i]; }

}

static public Matrix operator ~(Matrix m)

{

double[][] \_NewMatrix = new double[m.\_cols][];

for (uint i = 0; i < m.\_cols; i++)

{

\_NewMatrix[i] = new double[m.\_rows];

for (uint j = 0; j < m.\_rows; j++)

{

\_NewMatrix[i][j] = m.\_Matrix[j][i];

}

}

m.\_Matrix = \_NewMatrix;

uint tmp = m.\_rows; //swap

m.\_rows = m.\_cols;

m.\_cols = tmp;

return m;

}

static public Matrix operator +(Matrix m1, Matrix m2)

{

if(m1.\_rows != m2.\_rows || m1.\_cols != m2.\_cols)

{

throw new ArgumentException();

}

Matrix tmp = new Matrix(m1.\_rows, m1.\_cols);

for (uint i = 0; i < m1.\_rows; i++)

{

for (uint j = 0; j < m1.\_cols; j++)

{

tmp.\_Matrix[i][j] = m1.\_Matrix[i][j] + m2.\_Matrix[i][j];

}

}

return tmp;

}

static public Matrix operator -(Matrix m1, Matrix m2)

{

if (m1.\_rows != m2.\_rows || m1.\_cols != m2.\_cols)

{

throw new ArgumentException();

}

Matrix tmp = new Matrix(m1.\_rows, m1.\_cols);

for (uint i = 0; i < m1.\_rows; i++)

{

for (uint j = 0; j < m1.\_cols; j++)

{

tmp.\_Matrix[i][j] = m1.\_Matrix[i][j] - m2.\_Matrix[i][j];

}

}

return tmp;

}

static public Matrix operator \*(Matrix m1, Matrix m2)

{

if (m1.\_cols != m2.\_rows)

{

throw new ArgumentException();

}

uint i, j, k;

Matrix tmp = new Matrix(m1.\_rows, m2.\_cols);

for (i = 0; i < m1.\_rows; i++)

{

for (j = 0; j < m2.\_cols; j++)

{

tmp.\_Matrix[i][j] = 0;

for (k = 0; k < m1.\_cols; k++)

{

tmp.\_Matrix[i][j] += m1.\_Matrix[i][k] \* m2.\_Matrix[k][j];

}

}

}

return tmp;

}

public override string ToString()

{

string s = "";

uint i = 0, j;

for (; i < \_rows; i++)

{

j = 0;

for (; j < \_cols; j++)

{

s += \_Matrix[i][j];

s += ' ';

}

s += Environment.NewLine;

}

return s;

}

public Matrix(uint rows, uint cols)

{

\_rows = rows;

\_cols = cols;

\_Matrix = new double[\_rows][];

for (uint i = 0; i < \_rows; i++)

{

\_Matrix[i] = new double[\_cols];

}

}

protected uint \_rows;

protected uint \_cols;

protected double[][] \_Matrix = null;

}

}

SquareMatrices.cs:

namespace SquareMatrices

{

using Matrices;

class SquareMatrix : Matrix

{

static public SquareMatrix operator ~(SquareMatrix m)

{

double tmp;

for (uint i = 0; i < m.\_rows; i++)

{

for (uint j = 0; j < m.\_cols; j++)

{

tmp = m.\_Matrix[j][i]; //swap

m.\_Matrix[j][i] = m.\_Matrix[i][j];

m.\_Matrix[i][j] = tmp;

}

}

return m.Clone();

}

public SquareMatrix(uint rowcols)

: base(rowcols, rowcols) { }

public override double Det()

{

double D = 0;

uint n = GetRows();

if (n == 0)

{

return 0;

}

if (n == 1)

{

return this[0][0];

}

SquareMatrix tmp = new SquareMatrix(n - 1);

int sign = 1;

for (uint f = 0; f < n; f++)

{

GetCofactor(this, ref tmp, 0, f);

D += sign \* this[0][f] \* tmp.Det();

sign = -sign;

}

return D;

}

private void GetCofactor

(SquareMatrix thisM, ref SquareMatrix tmp, uint excessRow, uint excessCol)

{

uint i = 0, j = 0, n = thisM.GetRows();

for (uint row = 0; row < n; row++)

{

for (uint col = 0; col < n; col++)

{

if (row != excessRow && col != excessCol)

{

tmp[i][j++] = thisM[row][col];

if (j == n - 1)

{

j = 0;

i++;

}

}

}

}

}

public virtual SquareMatrix Clone()

{

SquareMatrix result = new SquareMatrix(\_rows);

for (uint i = 0; i < \_rows; i++)

{

for (uint j = 0; j < \_rows; j++)

{

result.\_Matrix[i][j] = \_Matrix[i][j];

}

}

return result;

}

}

}

FactoryMatrix.cs:

namespace Factory

{

using Matrices;

using SquareMatrices;

class FactoryMatrix

{

static public Matrix CreateMatrix(uint rows, uint cols)

{

if(rows==cols)

{

return new SquareMatrix(rows);

}

return new Matrix(rows, cols);

}

}

}

Form1.cs:

using System;

using System.Windows.Forms;

namespace Lab5\_Serchenko

{

using Factory;

using Matrices;

using System.Text.RegularExpressions;

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void add\_Click(object sender, EventArgs e)

{

try

{

string m1Text = Convert.ToString(matrix\_1.Text);

Matrix m1 = ScanMatrix(m1Text);

string m2Text = Convert.ToString(matrix\_2.Text);

Matrix m2 = ScanMatrix(m2Text);

Matrix m3 = m1 + m2;

matrix\_3.Text = m3.ToString();

errorText.Text = "";

}

catch(Exception exc)

{

errorText.Text = exc.Message;

}

}

private void sub\_Click(object sender, EventArgs e)

{

try

{

string m1Text = Convert.ToString(matrix\_1.Text);

Matrix m1 = ScanMatrix(m1Text);

string m2Text = Convert.ToString(matrix\_2.Text);

Matrix m2 = ScanMatrix(m2Text);

Matrix m3 = m1 - m2;

matrix\_3.Text = m3.ToString();

errorText.Text = "";

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private void mul\_Click(object sender, EventArgs e)

{

try

{

string m1Text = Convert.ToString(matrix\_1.Text);

Matrix m1 = ScanMatrix(m1Text);

string m2Text = Convert.ToString(matrix\_2.Text);

Matrix m2 = ScanMatrix(m2Text);

Matrix m3 = m1 \* m2;

matrix\_3.Text = m3.ToString();

errorText.Text = "";

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private void tra\_1\_Click(object sender, EventArgs e)

{

try

{

string m1Text = Convert.ToString(matrix\_1.Text);

Matrix m1 = ScanMatrix(m1Text);

\_ = ~m1;

matrix\_1.Text = m1.ToString();

errorText.Text = "";

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private void tra\_2\_Click(object sender, EventArgs e)

{

try

{

string m2Text = Convert.ToString(matrix\_2.Text);

Matrix m2 = ScanMatrix(m2Text);

\_ = ~m2;

matrix\_2.Text = m2.ToString();

errorText.Text = "";

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private void det\_1\_Click(object sender, EventArgs e)

{

try

{

string m1Text = Convert.ToString(matrix\_1.Text);

Matrix m1 = ScanMatrix(m1Text);

det\_res\_1.Text = m1.Det().ToString();

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private void det\_2\_Click(object sender, EventArgs e)

{

try

{

string m2Text = Convert.ToString(matrix\_2.Text);

Matrix m2 = ScanMatrix(m2Text);

det\_res\_2.Text = m2.Det().ToString();

}

catch (Exception exc)

{

errorText.Text = exc.Message;

}

}

private Matrix ScanMatrix(string text)

{

uint rows = 0, cols = 0;

foreach (var row in Regex.Split(text, "\r\n|\r|\n"))

{

if (row != "")

{

cols = 0;

foreach (var col in row.Trim().Split(' '))

{

cols++;

}

rows++;

}

}

Matrix m = FactoryMatrix.CreateMatrix(rows, cols);

rows = 0;

foreach (var row in Regex.Split(text, "\r\n|\r|\n"))

{

if (row != "")

{

cols = 0;

foreach (var col in row.Trim().Split(' '))

{

if (double.TryParse(col, out m[rows][cols++]) == false)

{

throw new ArgumentException();

}

}

rows++;

}

}

return m;

}

}

}

Program.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab5\_Serchenko

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Form1.Designer.cs:

namespace Lab5\_Serchenko

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.add = new System.Windows.Forms.Button();

this.matrix\_1 = new System.Windows.Forms.TextBox();

this.matrix\_2 = new System.Windows.Forms.TextBox();

this.sub = new System.Windows.Forms.Button();

this.mul = new System.Windows.Forms.Button();

this.matrix\_3 = new System.Windows.Forms.TextBox();

this.label2 = new System.Windows.Forms.Label();

this.det\_1 = new System.Windows.Forms.Button();

this.tra\_1 = new System.Windows.Forms.Button();

this.det\_res\_1 = new System.Windows.Forms.Label();

this.det\_res\_2 = new System.Windows.Forms.Label();

this.det\_2 = new System.Windows.Forms.Button();

this.tra\_2 = new System.Windows.Forms.Button();

this.errorText = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// add

//

this.add.Location = new System.Drawing.Point(378, 158);

this.add.Name = "add";

this.add.Size = new System.Drawing.Size(30, 30);

this.add.TabIndex = 0;

this.add.Text = "+";

this.add.UseVisualStyleBackColor = true;

this.add.Click += new System.EventHandler(this.add\_Click);

//

// matrix\_1

//

this.matrix\_1.Location = new System.Drawing.Point(77, 110);

this.matrix\_1.Multiline = true;

this.matrix\_1.Name = "matrix\_1";

this.matrix\_1.Size = new System.Drawing.Size(200, 200);

this.matrix\_1.TabIndex = 1;

//

// matrix\_2

//

this.matrix\_2.Location = new System.Drawing.Point(517, 110);

this.matrix\_2.Multiline = true;

this.matrix\_2.Name = "matrix\_2";

this.matrix\_2.Size = new System.Drawing.Size(200, 200);

this.matrix\_2.TabIndex = 2;

//

// sub

//

this.sub.Location = new System.Drawing.Point(378, 194);

this.sub.Name = "sub";

this.sub.Size = new System.Drawing.Size(30, 30);

this.sub.TabIndex = 3;

this.sub.Text = "-";

this.sub.UseVisualStyleBackColor = true;

this.sub.Click += new System.EventHandler(this.sub\_Click);

//

// mul

//

this.mul.Location = new System.Drawing.Point(378, 230);

this.mul.Name = "mul";

this.mul.Size = new System.Drawing.Size(30, 30);

this.mul.TabIndex = 4;

this.mul.Text = "\*";

this.mul.UseVisualStyleBackColor = true;

this.mul.Click += new System.EventHandler(this.mul\_Click);

//

// matrix\_3

//

this.matrix\_3.Location = new System.Drawing.Point(823, 110);

this.matrix\_3.Multiline = true;

this.matrix\_3.Name = "matrix\_3";

this.matrix\_3.Size = new System.Drawing.Size(200, 200);

this.matrix\_3.TabIndex = 6;

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(758, 199);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(18, 20);

this.label2.TabIndex = 8;

this.label2.Text = "=";

//

// det\_1

//

this.det\_1.AutoSize = true;

this.det\_1.Location = new System.Drawing.Point(112, 338);

this.det\_1.Name = "det\_1";

this.det\_1.Size = new System.Drawing.Size(42, 30);

this.det\_1.TabIndex = 10;

this.det\_1.Text = "det";

this.det\_1.UseVisualStyleBackColor = true;

this.det\_1.Click += new System.EventHandler(this.det\_1\_Click);

//

// tra\_1

//

this.tra\_1.AutoSize = true;

this.tra\_1.Location = new System.Drawing.Point(21, 199);

this.tra\_1.Name = "tra\_1";

this.tra\_1.Size = new System.Drawing.Size(42, 30);

this.tra\_1.TabIndex = 9;

this.tra\_1.Text = "~";

this.tra\_1.UseVisualStyleBackColor = true;

this.tra\_1.Click += new System.EventHandler(this.tra\_1\_Click);

//

// det\_res\_1

//

this.det\_res\_1.AutoSize = true;

this.det\_res\_1.Location = new System.Drawing.Point(201, 343);

this.det\_res\_1.Name = "det\_res\_1";

this.det\_res\_1.Size = new System.Drawing.Size(18, 20);

this.det\_res\_1.TabIndex = 11;

this.det\_res\_1.Text = "0";

//

// det\_res\_2

//

this.det\_res\_2.AutoSize = true;

this.det\_res\_2.Location = new System.Drawing.Point(644, 343);

this.det\_res\_2.Name = "det\_res\_2";

this.det\_res\_2.Size = new System.Drawing.Size(18, 20);

this.det\_res\_2.TabIndex = 15;

this.det\_res\_2.Text = "0";

//

// det\_2

//

this.det\_2.AutoSize = true;

this.det\_2.Location = new System.Drawing.Point(555, 338);

this.det\_2.Name = "det\_2";

this.det\_2.Size = new System.Drawing.Size(42, 30);

this.det\_2.TabIndex = 14;

this.det\_2.Text = "det";

this.det\_2.UseVisualStyleBackColor = true;

this.det\_2.Click += new System.EventHandler(this.det\_2\_Click);

//

// tra\_2

//

this.tra\_2.AutoSize = true;

this.tra\_2.Location = new System.Drawing.Point(458, 194);

this.tra\_2.Name = "tra\_2";

this.tra\_2.Size = new System.Drawing.Size(42, 30);

this.tra\_2.TabIndex = 16;

this.tra\_2.Text = "~";

this.tra\_2.UseVisualStyleBackColor = true;

this.tra\_2.Click += new System.EventHandler(this.tra\_2\_Click);

//

// errorText

//

this.errorText.AutoSize = true;

this.errorText.Location = new System.Drawing.Point(73, 491);

this.errorText.Name = "errorText";

this.errorText.Size = new System.Drawing.Size(0, 20);

this.errorText.TabIndex = 17;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1072, 677);

this.Controls.Add(this.errorText);

this.Controls.Add(this.tra\_2);

this.Controls.Add(this.det\_res\_2);

this.Controls.Add(this.det\_2);

this.Controls.Add(this.det\_res\_1);

this.Controls.Add(this.det\_1);

this.Controls.Add(this.tra\_1);

this.Controls.Add(this.label2);

this.Controls.Add(this.matrix\_3);

this.Controls.Add(this.mul);

this.Controls.Add(this.sub);

this.Controls.Add(this.matrix\_2);

this.Controls.Add(this.matrix\_1);

this.Controls.Add(this.add);

this.Name = "Form1";

this.Text = "Matrix calculator";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button add;

private System.Windows.Forms.TextBox matrix\_1;

private System.Windows.Forms.TextBox matrix\_2;

private System.Windows.Forms.Button sub;

private System.Windows.Forms.Button mul;

private System.Windows.Forms.TextBox matrix\_3;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Button det\_1;

private System.Windows.Forms.Button tra\_1;

private System.Windows.Forms.Label det\_res\_1;

private System.Windows.Forms.Label det\_res\_2;

private System.Windows.Forms.Button det\_2;

private System.Windows.Forms.Button tra\_2;

public System.Windows.Forms.Label errorText;

}

}