Національний технічний університет України

“Київський політехнічний інститут”

кафедра Технічної кібернетики

**Розробка програми для роботи з матрицями**

UA.IK.34020–01 01

*Курсова робота*

дисципліна

*“Об’єктно–орієнтоване програмування”*

Керівник: Виконавець:

доц. Лісовиченко О.І. Гарматін В. Д.

“Захист дозволено” гр. ІК-34

“\_\_\_” \_\_\_\_\_\_\_\_\_\_2014 р.

Захищено з оцінкою

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра Технічної кібернетики

Дисципліна Об‘єктно-орієнтоване програмування

Спеціальність 6.050201 «Комп’ютеризовані та робототехнічні системи»

Курс **I** Група **ІК-34** Семестр **II**

# ЗАВДАННЯ

**на курсову роботу студента**

Гарматіна Вячеслава Дмитровича

(прізвище, ім’я, по батькові)

1. Тема роботи Програма для роботи з матрицями

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи C#

4. Зміст розрахунково – пояснювальної записки (перелік питань, які підлягають розробці)

Розробка алгоритму програми

#### Архітектура програми

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень)

*Блок-схема алгоритму роботи програми (Формат А1)*

*Блок-схема додаткових ф-цій (Формат А1)*

6. Дата видачі завдання 25.04.2014

## КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  п\п | **Назва етапів курсової роботи** | Строк виконання **етапів роботи** | Примітки |
| **1** | Визначення теми | 25.04.2014 |  |
| **2** | Обробка літератури по темі | 28.04.2014 |  |
| **3** | Створення структури програми | 01.05.2014 |  |
| **4** | Визначення попереднього алгоритму | 02.05.2014 |  |
| **5** | Розробка попереднього інтерфейсу | 20.05.2014 |  |
| **6** | Написання основного коду програми | 20.05.2014 |  |
| **7** | Оптимізація і тестування | 25.05.2014 |  |
| **8** | Наладка програми та фінальна компіляція | 30.05.2014 |  |
| **9** | Оформлення роботи | 03.06.2014 |  |
| **10** | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент Гарматін В. Д.

(прізвище, ім’я, по батькові студента)

Керівник Лісовиченко О.І.

( прізвище, ім’я, по батькові викладача)

« » 2014 р.

**Зміст**

1. Вступ……………………………………………………………………………….………………………………….2
2. Постановка задачі.……………………………………………………………….……………………..……..4
3. Обґрунтування і вибір алгоритму……………………………………………………………………..5
4. Розробка програми……………………………………………………………………………………….…..7
   1. Загальні відомості……………………………………………….………...…
   2. Функціональне призначення…………………………………..………..…..
   3. Опис логічної структури програми………………………………………...
5. Керівництво користувача………………………………………………………………………..………...9
6. Керівництво розробника ….…………………………………………………………………….……….12
7. Отримані результати………………………………………………..……………………………………...13
   1. Особливості трансляції, компонування і налагодження………..…………
   2. Результати контрольного прикладу…………………………………..…….
8. Висновок ………………………………………………….………………………………………………………..16
9. **Використана література**…………………………………………………………………………..……..17

Додаток 1 (лістинг програми)……………………………………………………………………..…..…

Додаток 2 (блок-схема)………………………………………………………………………….……...…..

Додаток 3 (CD - диск)………………………………………………………………………….…….…….….

1. **Вступ**

Матриця — математичний об'єкт, записаний у вигляді прямокутної таблиці чисел (чи елементів кільця), він допускає операції (додавання, віднімання, множення та множення на число). Зазвичай матриці представляються двовимірними (прямокутними) таблицями. Іноді розглядають багатовимірні матриці або матриці непрямокутної форми. Матриці є корисними для запису даних, що залежать від двох категорій, наприклад: для коефіцієнтів систем лінійних рівнянь та лінійних перетворень.

Матрицею розміру m x n (m-на-n, або mn-матрицею) називається множина з елементів , розміщених у вигляді прямокутної таблиці з m рядків і n стовпців, а m і n — її розмірністю:

 \mathbf{A} = 
 \begin{bmatrix}
 a_{11} & a_{12} & \cdots & a_{1n} \\
 a_{21} & a_{22} & \cdots & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{m1} & a_{m2} & \cdots & a_{mn}
 \end{bmatrix}.


де a_{i,j}– елемент матриці; i– номер рядка; j– номер стовпця.

Горизонтальні лінії в матриці звуть рядками, вертикальні — стовпцями. Елемент матриці A, що знаходиться на перетині i-го рядка з j-им стовпчиком, називають i,j-им елементом або (i,j)-им елементом A.

Записують це як a_{i,j} чи a[i,j], або, в нотації мови програмування C, A[i][j]. Часто пишуть \ A:=(a_{i,j})_{n \times m} для означення матриці A розмірності n x m, де кожен елемент матриці A[i,j] позначають як a_{i,j} для всіх 1 ≤ i ≤ n та 1 ≤ j ≤ m.

Метою курсової роботи є створення програми яка буде проводити основні дії над матрицями, такі як додавання, множення на скаляр, додавання та множення матриць, зведення у степінь та транспонування.

1. **Постановка задачі**

Реалізувати програму з графічним інтерфейсом в якому реалізувати:

- Дії над матрицями:

Транспонування матриці;

Знаходження оберненої матриці;

Множення матриці на число;

Множення матриці на матрицю;

Знаходження визначника;

Додавання матриці;

Зведення матриці в степінь;

- Збереження та завантаження матриць;

1. **Обґрунтування і вибір алгоритму**

***Алгоритм додавання матриць***

Якщо дано дві матриці m-на-n A і B, можемо визначити їх суму A + B як матрицю m-на-n, що утворюється додаванням відповідних елементів, себто, (A + B)[i, j] = A[i, j] + B[i, j].

Наприклад:


  \begin{bmatrix}
    1 & 3 & 2 \\
    1 & 0 & 0 \\
    1 & 2 & 2
  \end{bmatrix}
+
  \begin{bmatrix}
    0 & 0 & 5 \\
    7 & 5 & 0 \\
    2 & 1 & 1
  \end{bmatrix}
=
  \begin{bmatrix}
    1+0 & 3+0 & 2+5 \\
    1+7 & 0+5 & 0+0 \\
    1+2 & 2+1 & 2+1
  \end{bmatrix}
=
  \begin{bmatrix}
    1 & 3 & 7 \\
    8 & 5 & 0 \\
    3 & 3 & 3
  \end{bmatrix}


***Алгоритм множення на скаляр(число)***

Якщо дано матрицю A і число c, можемо означити множення на скаляр cA як (cA)[i, j] = cA[i, j].

Наприклад:

2
  \begin{bmatrix}
    1 & 8 & -3 \\
    4 & -2 & 5
  \end{bmatrix}
=\begin{bmatrix}
    2\times 1 & 2\times 8 & 2\times -3 \\
    2\times 4 & 2\times -2 & 2\times 5
  \end{bmatrix}
=\begin{bmatrix}
    2 & 16 & -6 \\
    8 & -4 & 10
  \end{bmatrix}


***Алгоритм множення матриць***

Множення двох матриць має сенс лише тоді, коли число стовпчиків першої матриці дорівнює числу рядків другої матриці. Якщо A — матриця m-на-n (m рядків, n стовпчиків), а B — матриця n-на-p (n рядків, p стовпчиків), їх добуток AB є матрицею m-на-p (m рядків, p стовпчиків), що розраховується за формулою:

(AB)[i, j] = A[i, 1] \* B[1, j] + A[i, 2] \* B[2, j] + ... + A[i, n] \* B[n, j] для кожної пари i та j.

Наприклад:


  \begin{bmatrix}
    1 & 0 & 2 \\
    -1 & 3 & 1 \\
  \end{bmatrix}
\times
  \begin{bmatrix}
    3 & 1 \\
    2 & 1 \\
    1 & 0
  \end{bmatrix}
=\begin{bmatrix}
     (1 \times 3  +  0 \times 2  +  2 \times 1) & (1 \times 1   +   0 \times 1   +   2 \times 0) \\
    (-1 \times 3  +  3 \times 2  +  1 \times 1) & (-1 \times 1   +   3 \times 1   +   1 \times 0) \\
  \end{bmatrix}
=\begin{bmatrix}
    5 & 1 \\
    4 & 2 \\
  \end{bmatrix}


***Алгоритм транспонування***

Транспонована матриця — матриця A^T, що виникає з матриці A в результаті унарної операції транспонування: заміни її рядків на стовпчики.

Наприклад:

\begin{bmatrix}
1 & 2  \\
3 & 4 \end{bmatrix}^\mathrm{T}
= \begin{bmatrix}
1 & 3  \\
2 & 4 \end{bmatrix}
та 

\begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6 \end{bmatrix}^\mathrm{T}
= \begin{bmatrix}
1 & 3 & 5\\
2 & 4 & 6 \end{bmatrix}


1. **Розробка програми**
   1. Загальні відомості

Програма написана в Visual Studio на мові програмування C#.

***Системні вимоги:***

.Net Framework 4.5.1 або вище;

* 1. Функціональне призначення

Програма призначена для виконання основних дій над матрицями.

* 1. Опис логічної структури програми

Програма реалізована у вигляді програми з графічним інтерфейсом:

Виведення програми в якій реалізовано:

1. Збереження та завантаження матриць та інших даних.
2. Генерування порожньої матриці заданого розміру.
3. Пункти меню, які відповідають основним діям над матрицями:

3.1 Транспонування;

3.2 Знаходження оберненої матриці;

3.3 Множення матриці на число;

3.4 Зведення матриці в степінь;

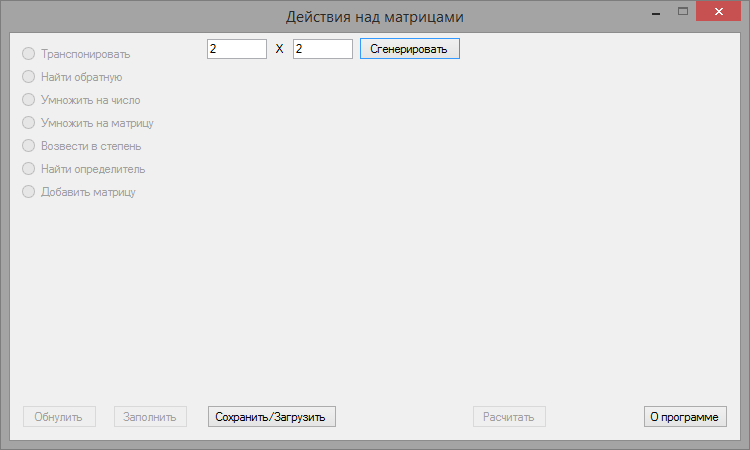
3.4 Знаходження визначника;

3.4 Множення матриці на матрицю;

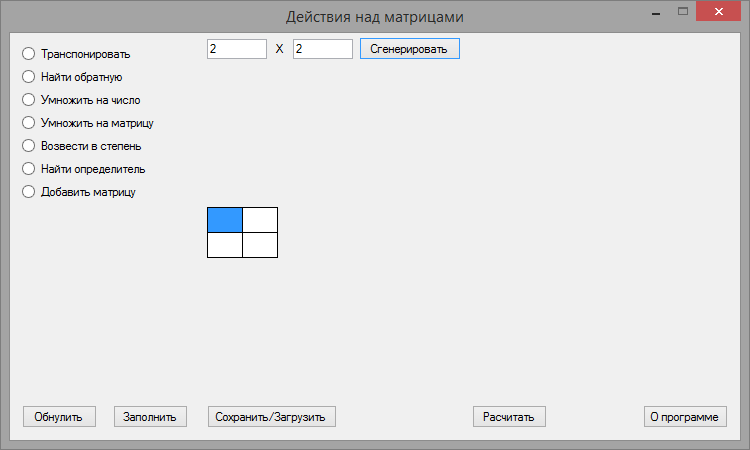
3.4 Додавання матриць;

1. **Керівництво користувача**

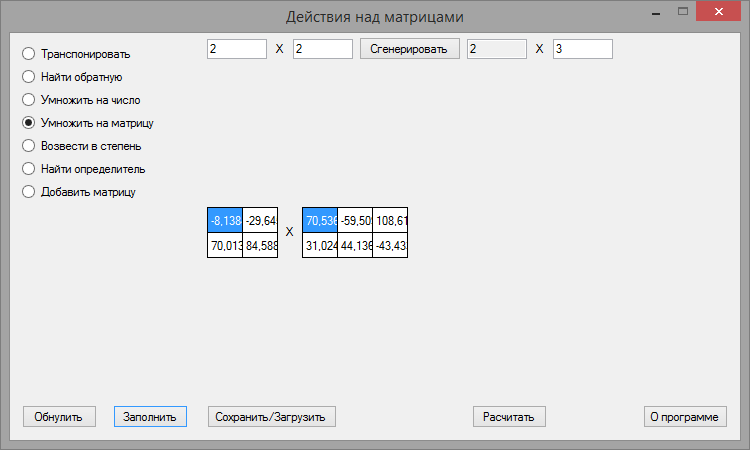
Після запуску програми перед користувачем з’явиться вікно програми:

****

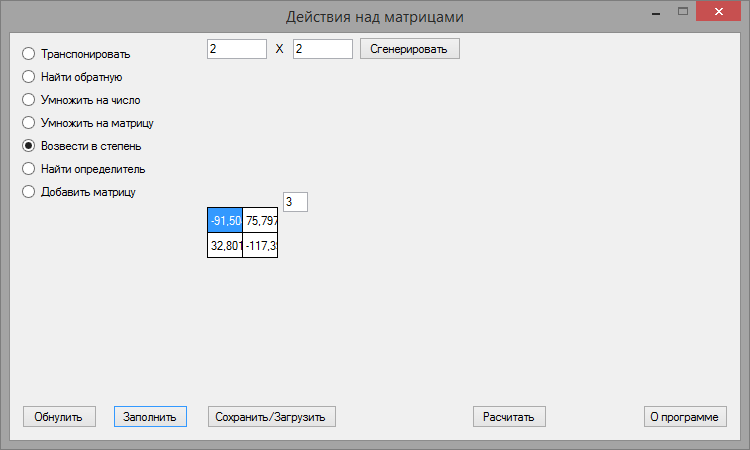
1. Для того щоб розпочати роботу з програмою треба згенерувати порожню матрицю, попередньо зазначивши розміри 1 матриці.



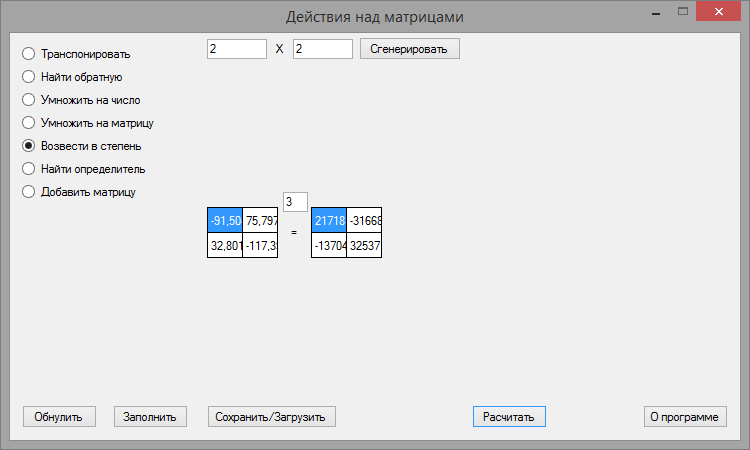
1. Після створення 1 матриці потрібно заповнити матрицю та вибрати потрібний пункт меню.

****

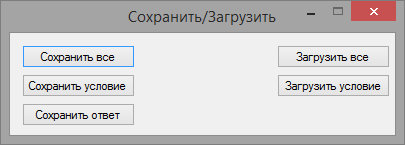
1. Якщо потрібно, згенерувати та заповнити 2 матрицю.



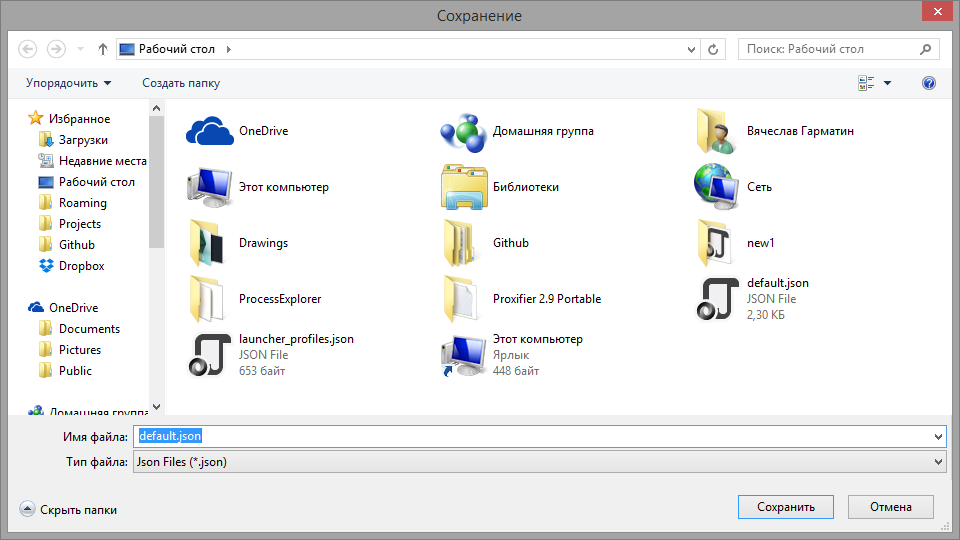
1. Або вказати додаткове число (степінь, яке число додавати або на яке множити).



1. Натиснути кнопку “Расчитать”.



1. Якщо потрібно, можна завантажити або зберегти отримані результати, натиснувши кнопку “Сохранить/Загрузить”.



1. Для збереження файлу, треба вказати що саме зберегти та вибрати назву файла. (По замовчуванню ім’я файлу default.json)
2. Для завантаження даних з файлу, треба вказати що саме завантажувати, та вказати назву файлу. (По замовчуванню ім’я файлу default.json)
3. **Керівництво розробника**

**Перелік класів, функцій та форм:**

1. **Form1 – головне вікно, де знаходяться всі пункти меню.**

**1.1 Клас Form1 – в ньому розташована основна програма**

**1.2 Клас utils – в ньмоу містятся ф-ції для обчислення оберненоі матриці         та визначника.**

**2. Form2 – додаткове вікно, для збереження та завантаження файлів.**

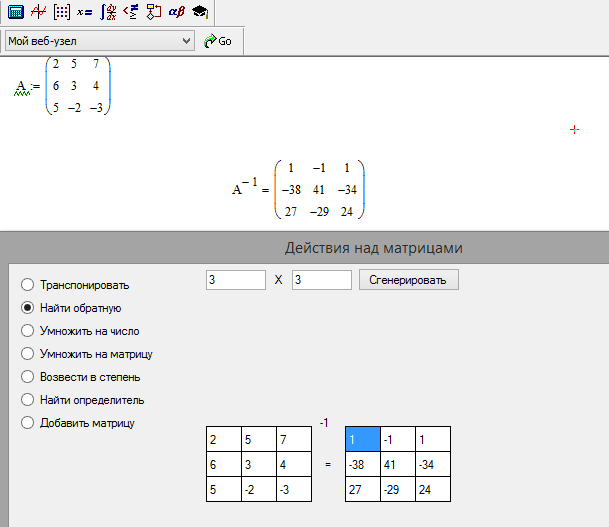
**3. Form 3 – додаткове вікно, де описана програма та розробник**

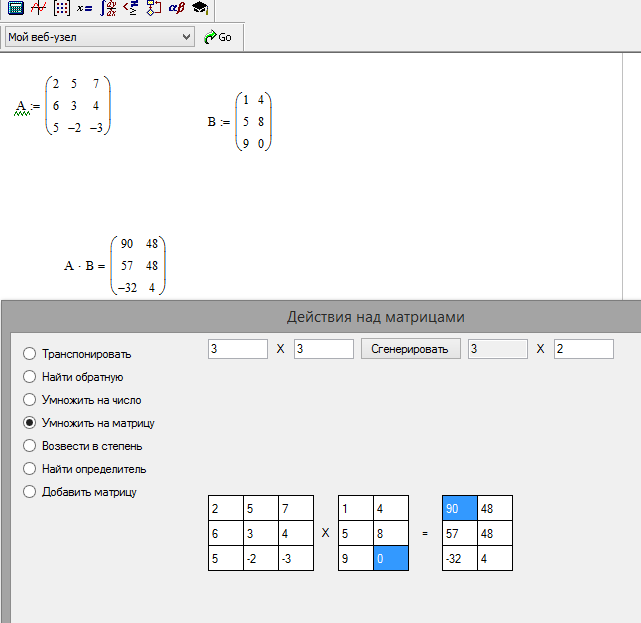
1. **Отримані результати**
   1. Особливості трансляції, компонування і налагодження

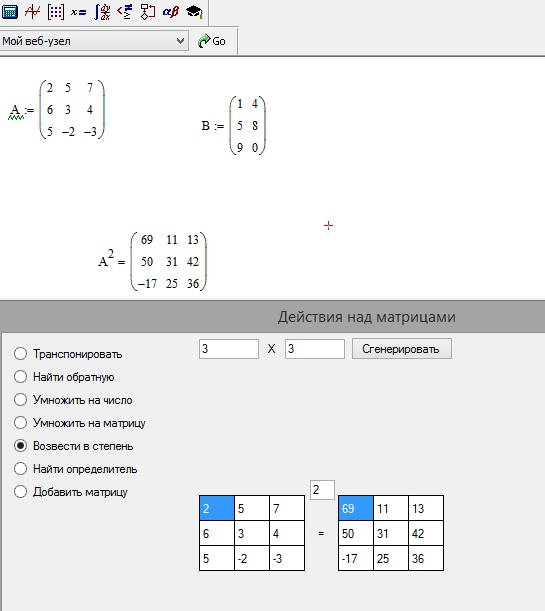
На початкових етапах розробки програми основна увага приділялась правильності роботи алгоритмів дій над матрицями. Відбувалося налагодження методів взаємодії інтерфейсу з фізичною частиною програми.

Найтривалішим етапом розробки програми будо створити правильний алгоритм для знаходження оберненої матриці.

* 1. Результати контрольного прикладу







1. **Висновок**

У результаті виконання курсової роботи була спроектована і реалізована проста і зручна в застосуванні програма.

**Переваги створеної програми:**

* Простота у використанні;
* Програма використовує небагато комп’ютерних ресурсів;
* Створення файлу, який можна редагувати.

1. **Використана література**
2. Wikipedia
3. MSDN

**Додаток 1**

**Лістинг (код) програми**

Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kursach2\_WF\_

{

public partial class Form1 : Form

{

private int opt = 0;

private int N, M;

private int n, m;

private bool check;

private double[,] values;

private double[,] values2;

private double[,] svalues;

private string number;

private bool loading;

private string[,] firstm;

private string[,] answer;

private string[,] secondm;

public Form1()

{

InitializeComponent();

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8))

{

e.Handled = true;

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8))

{

e.Handled = true;

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (opt == 3)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && e.KeyChar != '-' && e.KeyChar != ',')

{

e.Handled = true;

}

if (e.KeyChar == '-' && (sender as TextBox).Text.Length > 0)

{

e.Handled = true;

}

if (e.KeyChar == ',' && (sender as TextBox).Text.IndexOf(',') > -1)

{

e.Handled = true;

}

}

else if (opt == 5)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8))

{

e.Handled = true;

}

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && e.KeyChar != Convert.ToChar(8))

{

e.Handled = true;

}

}

private void dataGridView1\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

TextBox tb = e.Control as TextBox;

if (tb != null)

{

e.Control.KeyPress += new KeyPressEventHandler(dataGridView1\_KeyPress);

}

}

private void dataGridView1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && e.KeyChar != '-' && e.KeyChar != ',')

{

e.Handled = true;

}

if (e.KeyChar == '-' && (sender as TextBox).Text.Length > 0)

{

e.Handled = true;

}

if (e.KeyChar == ',' && (sender as TextBox).Text.IndexOf(',') > -1)

{

e.Handled = true;

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBox1.Text) < 1 || Convert.ToInt32(textBox1.Text) > 10)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

if (opt == 7)

{

textBox5.Text = textBox1.Text;

}

}

}

catch

{

button1.Enabled = false;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBox2.Text) < 1 || Convert.ToInt32(textBox2.Text) > 10)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

if (opt == 7)

{

textBox4.Text = textBox2.Text;

}

else if (opt == 4)

{

textBox5.Text = textBox2.Text;

}

}

}

catch

{

button1.Enabled = false;

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

number = textBox3.Text;

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBox4.Text) < 1 || Convert.ToInt32(textBox4.Text) > 10)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

}

}

catch

{

button1.Enabled = false;

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label3.Visible = false;

textBox3.Visible = false;

textBox3.Text = "";

label4.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

dataGridView1.Visible = true;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

textBox4.ReadOnly = false;

opt = 1;

datagridsize(false);

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X + dataGridView1.Size.Width + 5);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y - 10);

label2.Text = "T";

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label3.Visible = false;

textBox3.Visible = false;

textBox3.Text = "";

label4.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

dataGridView1.Visible = true;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

textBox4.ReadOnly = false;

opt = 2;

datagridsize(false);

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X + dataGridView1.Size.Width + 5);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y - 10);

label2.Text = "-1";

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label2.Text = "X";

label3.Visible = false;

textBox3.Visible = true;

textBox3.Text = "";

label4.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

dataGridView1.Visible = true;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

textBox4.ReadOnly = false;

opt = 3;

datagridsize(false);

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X - label2.Size.Width - 3);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 7);

textBox3.Location = new Point(this.textBox3.Location.X, dataGridView1.Location.X - textBox3.Size.Width - label2.Size.Width - 6);

textBox3.Location = new Point(this.textBox3.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 10);

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label2.Text = "X";

label3.Visible = true;

textBox3.Visible = false;

textBox4.Visible = true;

textBox5.Visible = true;

label4.Visible = false;

textBox4.ReadOnly = false;

textBox5.Text = textBox2.Text;

dataGridView1.Visible = true;

dataGridView2.Visible = false;

opt = 4;

datagridsize(true);

if (check == false)

{

button1.PerformClick();

check = true;

}

int z = dataGridView1.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

int y = dataGridView1.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (z > 126 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

z = 126;

}

if (y > 101 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

z = z + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X + z + 5);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 7);

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = false;

label3.Visible = false;

textBox3.Visible = true;

textBox3.Text = "";

label4.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

textBox4.ReadOnly = false;

dataGridView1.Visible = true;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

opt = 5;

datagridsize(false);

textBox3.Location = new Point(this.textBox3.Location.X, dataGridView1.Location.X + dataGridView1.Size.Width + 5);

textBox3.Location = new Point(this.textBox3.Location.Y, dataGridView1.Location.Y - 15);

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label3.Visible = false;

label2.Text = "det";

textBox3.Visible = false;

label4.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

textBox4.ReadOnly = false;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

opt = 6;

datagridsize(false);

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X - label2.Size.Width - 3);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 7);

}

private void radioButton7\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

label2.Text = "+";

label3.Visible = true;

textBox3.Visible = false;

label4.Visible = false;

textBox4.Visible = true;

textBox5.Visible = true;

textBox4.Text = textBox2.Text;

textBox5.Text = textBox1.Text;

textBox4.ReadOnly = true;

dataGridView2.Visible = false;

opt = 7;

datagridsize(true);

if (check == false)

{

button1.PerformClick();

check = true;

}

int z = dataGridView1.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

int y = dataGridView1.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (z > 126 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

z = 126;

}

if (y > 101 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

z = z + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

label2.Location = new Point(this.label2.Location.X, dataGridView1.Location.X + z + 5);

label2.Location = new Point(this.label2.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 7);

}

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (loading == false)

{

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

{

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

{

values[i, j] = Convert.ToDouble(dataGridView1.Rows[i].Cells[j].Value);

}

}

}

}

private void dataGridView3\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (loading == false)

{

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

{

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

{

svalues[i, j] = Convert.ToDouble(dataGridView3.Rows[i].Cells[j].Value);

}

}

}

}

private void dataGridView2\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

radioButton1.Enabled = false;

radioButton2.Enabled = false;

radioButton3.Enabled = false;

radioButton4.Enabled = false;

radioButton5.Enabled = false;

radioButton6.Enabled = false;

radioButton7.Enabled = false;

label4.Visible = false;

label3.Visible = false;

label2.Visible = false;

button2.Enabled = false;

button3.Enabled = false;

button5.Enabled = false;

textBox4.Visible = false;

textBox5.Visible = false;

dataGridView2.Visible = false;

dataGridView3.Visible = false;

dataGridView1.Visible = true;

opt = 0;

M = Convert.ToInt32(textBox1.Text);

N = Convert.ToInt32(textBox2.Text);

dataGridView1.RowCount = M;

dataGridView1.ColumnCount = N;

int i = 0, j = 0;

if (loading == false)

{

values = new double[M, N];

}

for (i = 0; i < M; i++)

{

for (j = 0; j < N; j++)

{

dataGridView1.Columns[j].Width = 35;

dataGridView1.Rows[i].Height = 25;

((DataGridViewTextBoxColumn)dataGridView1.Columns[j]).MaxInputLength = 10;

}

}

m = Convert.ToInt32(textBox5.Text);

n = Convert.ToInt32(textBox4.Text);

dataGridView3.RowCount = m;

dataGridView3.ColumnCount = n;

if (loading == false)

{

svalues = new double[m, n];

}

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

dataGridView3.Columns[j].Width = 35;

dataGridView3.Rows[i].Height = 25;

((DataGridViewTextBoxColumn)dataGridView3.Columns[j]).MaxInputLength = 10;

}

}

if (opt == 4 || opt == 7)

{

label3.Visible = true;

datagridsize(true);

}

else

{

datagridsize(false);

}

radioButton1.Enabled = true;

if (M == N)

{

radioButton2.Enabled = true;

radioButton5.Enabled = true;

radioButton6.Enabled = true;

}

radioButton3.Enabled = true;

radioButton4.Enabled = true;

radioButton7.Enabled = true;

button2.Enabled = true;

button3.Enabled = true;

button5.Enabled = true;

label2.Visible = false;

textBox3.Visible = false;

if (radioButton1.Checked == true)

{

radioButton1.Checked = false;

radioButton1.Checked = true;

}

if (radioButton2.Checked == true)

{

if (M == N)

{

radioButton2.Checked = false;

radioButton2.Checked = true;

}

else

{

radioButton2.Checked = false;

label2.Visible = false;

opt = 0;

}

}

if (radioButton3.Checked == true)

{

radioButton3.Checked = false;

radioButton3.Checked = true;

}

if (radioButton4.Checked == true)

{

radioButton4.Checked = false;

radioButton4.Checked = true;

dataGridView3.Visible = true;

}

if (radioButton5.Checked == true)

{

if (M == N)

{

radioButton5.Checked = false;

radioButton5.Checked = true;

}

else

{

radioButton5.Checked = false;

textBox3.Visible = false;

opt = 0;

}

}

if (radioButton6.Checked == true)

{

if (M == N)

{

radioButton6.Checked = false;

radioButton6.Checked = true;

}

else

{

radioButton6.Checked = false;

label2.Visible = false;

opt = 0;

}

}

if (radioButton7.Checked == true)

{

radioButton7.Checked = false;

radioButton7.Checked = true;

dataGridView3.Visible = true;

}

check = false;

}

private void button2\_Click(object sender, EventArgs e)

{

bool notfin = false;

values = new double[dataGridView1.Rows.Count, dataGridView1.Columns.Count];

svalues = new double[dataGridView3.Rows.Count, dataGridView3.Columns.Count];

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

if (dataGridView1.Rows[i].Cells[j].Value == null || dataGridView1.Rows[i].Cells[j].Value.ToString() == "-" || dataGridView1.Rows[i].Cells[j].Value.ToString() == "," || dataGridView1.Rows[i].Cells[j].Value.ToString() == "-,")

{

notfin = true;

break;

}

else

{

values[i, j] = Convert.ToDouble(dataGridView1.Rows[i].Cells[j].Value);

}

}

if (notfin == true)

{

break;

}

}

if (opt == 4 || opt == 7)

{

if (dataGridView3.Visible == true)

{

for (int i = 0; i < dataGridView3.Rows.Count; i++)

{

for (int j = 0; j < dataGridView3.Columns.Count; j++)

{

if (dataGridView3.Rows[i].Cells[j].Value == null || dataGridView3.Rows[i].Cells[j].Value.ToString() == "-" || dataGridView3.Rows[i].Cells[j].Value.ToString() == "," || dataGridView3.Rows[i].Cells[j].Value.ToString() == "-,")

{

notfin = true;

break;

}

else

{

svalues[i, j] = Convert.ToDouble(dataGridView3.Rows[i].Cells[j].Value);

}

}

if (notfin == true)

{

break;

}

}

}

else

{

notfin = true;

}

}

if (notfin == true)

{

MessageBox.Show("Заполните матрицу");

}

else

{

if (opt == 1)

{

answersize();

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Rows[i].Cells[j].Value = values[j, i];

}

}

}

else if (opt == 2)

{

int n = dataGridView1.Rows.Count;

values2 = new double[dataGridView1.Rows.Count, dataGridView1.Columns.Count];

double det = utils.Determ(n, values);

if (det != 0)

{

answersize();

if (n > 1)

{

values2 = utils.Inverse(n, values);

}

else

{

values2 = values;

}

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Rows[i].Cells[j].Value = values2[i, j];

}

}

}

else

{

MessageBox.Show("У даной матрицы определитель 0. Невозможно найти матрицу с определителем 0");

}

}

else if (opt == 3)

{

if (textBox3.Text != "" && textBox3.Text != "-" && textBox3.Text != "-," && textBox3.Text != ",")

{

answersize();

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Rows[i].Cells[j].Value = Convert.ToDouble(textBox3.Text.ToString()) \* values[i, j];

}

}

}

else

{

MessageBox.Show("Введите число на которое умножается матрица");

}

}

else if (opt == 4)

{

answersize();

values2 = new double[M, n];

for (int row = 0; row < M; row++)

{

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

{

for (int inner = 0; inner < dataGridView3.Rows.Count; inner++)

{

values2[row, col] += values[row, inner] \* svalues[inner, col];

}

}

}

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

{

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

{

dataGridView2.Rows[i].Cells[j].Value = values2[i, j];

}

}

}

else if (opt == 5)

{

if (textBox3.Text != "" && Convert.ToInt32(textBox3.Text) > 0)

{

answersize();

values2 = new double[M, N];

if (Convert.ToInt32(textBox3.Text) > 1)

{

for (int k = 0; k < Convert.ToInt32(textBox3.Text) - 1; k++)

{

for (int row = 0; row < M; row++)

{

for (int col = 0; col < N; col++)

{

for (int inner = 0; inner < M; inner++)

{

values2[row, col] += values[row, inner] \* values[inner, col];

}

}

}

}

}

else

{

values2 = values;

}

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Rows[i].Cells[j].Value = values2[i, j];

}

}

}

else

{

MessageBox.Show("Введите степень матрицы");

}

}

else if (opt == 6)

{

answersize();

double det = utils.Determ(M, values);

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Columns[j].Width = 35;

dataGridView2.Rows[i].Height = 25;

dataGridView2.Rows[i].Cells[j].Value = det;

}

}

}

else if (opt == 7)

{

answersize();

values2 = new double[M, n];

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

{

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

{

values2[i, j] = values[i, j] + svalues[i, j];

}

}

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

{

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

{

dataGridView2.Rows[i].Cells[j].Value = values2[i, j];

}

}

}

else

{

MessageBox.Show("Выберите действие");

}

}

}

private void button3\_Click(object sender, EventArgs e)

{

label4.Visible = false;

dataGridView2.Visible = false;

textBox3.Text = "";

int i = 0, j = 0;

for (i = 0; i < M; i++)

{

for (j = 0; j < N; j++)

{

dataGridView1.Rows[i].Cells[j].Value = null;

}

}

if (opt == 7 || opt == 4)

{

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

dataGridView3.Rows[i].Cells[j].Value = null;

}

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

loading = true;

firstm = new string[dataGridView1.RowCount, dataGridView1.ColumnCount];

for (int i = 0; i < dataGridView1.RowCount; i++)

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

firstm[i, j] = Convert.ToString(dataGridView1.Rows[i].Cells[j].Value);

}

if (dataGridView2.Visible == true)

{

answer = new string[dataGridView2.RowCount, dataGridView2.ColumnCount];

for (int i = 0; i < dataGridView2.RowCount; i++)

for (int j = 0; j < dataGridView2.ColumnCount; j++)

{

answer[i, j] = Convert.ToString(dataGridView2.Rows[i].Cells[j].Value);

}

}

else

{

answer = null;

}

if (dataGridView3.Visible == true)

{

secondm = new string[dataGridView3.RowCount, dataGridView3.ColumnCount];

for (int i = 0; i < dataGridView3.RowCount; i++)

for (int j = 0; j < dataGridView3.ColumnCount; j++)

{

secondm[i, j] = Convert.ToString(dataGridView3.Rows[i].Cells[j].Value);

}

}

else

{

secondm = null;

}

if (textBox3.Visible == false) number = null;

Form2 oForm2 = new Form2(this.opt, this.firstm, this.answer, this.secondm, this.number);

oForm2.ShowDialog();

bool aim = oForm2.retaim;

if (aim == true)

{

opt = oForm2.retopt;

number = oForm2.retnumber;

firstm = oForm2.retvalues;

secondm = oForm2.retsvalues;

answer = oForm2.retvalues2;

bool load = oForm2.retloadall;

if (opt == 0)

{

if (radioButton1.Checked == true)

{

radioButton1.Checked = false;

}

if (radioButton2.Checked == true)

{

radioButton2.Checked = false;

}

if (radioButton3.Checked == true)

{

radioButton3.Checked = false;

}

if (radioButton4.Checked == true)

{

radioButton4.Checked = false;

}

if (radioButton5.Checked == true)

{

radioButton5.Checked = false;

}

if (radioButton6.Checked == true)

{

radioButton6.Checked = false;

}

if (radioButton7.Checked == true)

{

radioButton7.Checked = false;

}

opt = 0;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

button1.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

}

else if (opt == 1)

{

opt = 1;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

radioButton1.Checked = true;

button1.PerformClick();

radioButton1.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

if (load == true)

{

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 2)

{

opt = 2;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

radioButton2.Checked = true;

button1.PerformClick();

radioButton2.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

if (load == true)

{

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 3)

{

opt = 3;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

radioButton3.Checked = true;

button1.PerformClick();

radioButton3.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

textBox3.Text = number;

if (load == true)

{

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 4)

{

opt = 4;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

if (secondm != null)

{

textBox4.Text = Convert.ToString(secondm.GetLength(1));

}

radioButton4.Checked = true;

button1.PerformClick();

radioButton4.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

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

for (int j = 0; j < secondm.GetLength(1); j++)

{

dataGridView3.Rows[i].Cells[j].Value = secondm[i, j];

}

if (load == true)

{

M = firstm.GetLength(0);

n = secondm.GetLength(1);

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 5)

{

opt = 5;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

radioButton5.Checked = true;

button1.PerformClick();

radioButton5.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

textBox3.Text = number;

if (load == true)

{

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 6)

{

opt = 6;

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

radioButton6.Checked = true;

button1.PerformClick();

radioButton6.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

if (load == true)

{

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

else if (opt == 7)

{

textBox1.Text = Convert.ToString(firstm.GetLength(0));

textBox2.Text = Convert.ToString(firstm.GetLength(1));

if (secondm != null)

{

textBox4.Text = Convert.ToString(secondm.GetLength(1));

}

radioButton7.Checked = true;

button1.PerformClick();

radioButton7.PerformClick();

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

for (int j = 0; j < firstm.GetLength(1); j++)

{

dataGridView1.Rows[i].Cells[j].Value = firstm[i, j];

}

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

for (int j = 0; j < secondm.GetLength(1); j++)

{

dataGridView3.Rows[i].Cells[j].Value = secondm[i, j];

}

if (load == true)

{

M = firstm.GetLength(0);

n = secondm.GetLength(1);

answersize();

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

{

for (int j = 0; j < answer.GetLength(1); j++)

{

dataGridView2.Rows[i].Cells[j].Value = answer[i, j];

}

}

}

}

}

loading = false;

}

private void button5\_Click(object sender, EventArgs e)

{

label4.Visible = false;

dataGridView2.Visible = false;

textBox3.Text = "";

Random random = new Random();

if (opt == 3)

{

double randomNumber = 125 + random.NextDouble() \* (-250);

textBox3.Text = randomNumber.ToString();

}

else if (opt == 5)

{

int randomNumber = random.Next(1, 10);

textBox3.Text = randomNumber.ToString();

}

int i = 0, j = 0;

for (i = 0; i < M; i++)

{

for (j = 0; j < N; j++)

{

double randomNumber = 125 + random.NextDouble() \* (-250);

dataGridView1.Rows[i].Cells[j].Value = randomNumber;

}

}

if (opt == 4 || opt == 7)

{

if (dataGridView3.Visible == true)

{

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

double randomNumber = 125 + random.NextDouble() \* (-250);

dataGridView3.Rows[i].Cells[j].Value = randomNumber;

}

}

}

}

}

public void datagridsize(bool small)

{

int x, y;

x = dataGridView1.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

y = dataGridView1.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (small == true)

{

if (x > 126 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

x = 127;

y = y + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight;

}

if (y > 101 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

y = 102;

x = x + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

dataGridView1.Location = new Point(this.dataGridView1.Location.X, textBox1.Location.X);

dataGridView1.Location = new Point(this.dataGridView1.Location.Y, (400 - y) / 2);

dataGridView1.Size = new System.Drawing.Size(x, y);

int i, j;

i = dataGridView3.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

j = dataGridView3.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (i > 126 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

i = 127;

j = j + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight;

}

if (j > 101 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

j = 102;

i = i + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

dataGridView3.Location = new Point(this.dataGridView3.Location.X, dataGridView1.Location.X + dataGridView1.Size.Width + label2.Width + 10);

dataGridView3.Location = new Point(this.dataGridView3.Location.Y, (400 - j) / 2);

dataGridView3.Size = new System.Drawing.Size(i, j);

}

else

{

if (x > 201 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

x = 201;

y = y + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight;

}

if (y > 161 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

y = 161;

x = x + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

dataGridView1.Location = new Point(this.dataGridView1.Location.X, textBox1.Location.X);

dataGridView1.Location = new Point(this.dataGridView1.Location.Y, (400 - y) / 2);

dataGridView1.Size = new System.Drawing.Size(x, y);

}

}

private void answersize()

{

label4.Visible = true;

label4.Text = "=";

dataGridView2.Visible = true;

if (opt == 1)

{

dataGridView2.RowCount = dataGridView1.Columns.Count;

dataGridView2.ColumnCount = dataGridView1.Rows.Count;

}

else if (opt == 2 || opt == 3 || opt == 5)

{

dataGridView2.RowCount = dataGridView1.Rows.Count;

dataGridView2.ColumnCount = dataGridView1.Columns.Count;

}

else if (opt == 4 || opt == 7)

{

dataGridView2.RowCount = M;

dataGridView2.ColumnCount = n;

}

else if (opt == 6)

{

dataGridView2.RowCount = 1;

dataGridView2.ColumnCount = 1;

}

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

for (int j = 0; j < dataGridView2.Columns.Count; j++)

{

dataGridView2.Columns[j].Width = 35;

dataGridView2.Rows[i].Height = 25;

((DataGridViewTextBoxColumn)dataGridView2.Columns[j]).MaxInputLength = 10;

}

}

if (opt == 4 || opt == 7)

{

int x, y;

x = dataGridView2.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

y = dataGridView2.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (x > 126 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

x = 126;

y = y + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight;

}

if (y > 101 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

y = 102;

x = x + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

label4.Location = new Point(this.label4.Location.X, dataGridView3.Location.X + dataGridView3.Size.Width + 10);

label4.Location = new Point(this.label4.Location.Y, dataGridView3.Location.Y + (dataGridView3.Size.Height / 2) - 7);

dataGridView2.Location = new Point(this.dataGridView2.Location.X, label4.Location.X + label4.Size.Width + 10);

dataGridView2.Location = new Point(this.dataGridView2.Location.Y, (400 - y) / 2);

dataGridView2.Size = new System.Drawing.Size(x, y);

}

else

{

int x, y;

x = dataGridView2.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) + 1;

y = dataGridView2.Rows.GetRowsHeight(DataGridViewElementStates.Visible) + 1;

if (x > 201 + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth)

{

x = 202;

y = y + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight;

}

if (y > 161 + System.Windows.Forms.SystemInformation.HorizontalScrollBarHeight)

{

y = 162;

x = x + System.Windows.Forms.SystemInformation.VerticalScrollBarWidth;

}

label4.Location = new Point(this.label4.Location.X, dataGridView1.Location.X + dataGridView1.Size.Width + 10);

label4.Location = new Point(this.label4.Location.Y, dataGridView1.Location.Y + (dataGridView1.Size.Height / 2) - 7);

dataGridView2.Location = new Point(this.dataGridView2.Location.X, label4.Location.X + label4.Size.Width + 10);

dataGridView2.Location = new Point(this.dataGridView2.Location.Y, (400 - y) / 2);

dataGridView2.Size = new System.Drawing.Size(x, y);

}

}

private void button6\_Click(object sender, EventArgs e)

{

Form3 oform3 = new Form3();

oform3.ShowDialog();

}

}

public class utils

{

public static double Determ(int n, double[,] values)

{

int i, j, j1, j2;

double det = 0;

double[,] values2 = new double[n - 1, n - 1];

if (n == 1)

{

det = values[0, 0];

}

else if (n == 2)

{

det = values[0, 0] \* values[1, 1] - values[1, 0] \* values[0, 1];

}

else

{

det = 0;

for (j1 = 0; j1 < n; j1++)

{

for (i = 1; i < n; i++)

{

j2 = 0;

for (j = 0; j < n; j++)

{

if (j == j1)

continue;

values2[i - 1, j2] = values[i, j];

j2++;

}

}

det += Math.Pow(-1.0, j1 + 2.0) \* values[0, j1] \* Determ(n - 1, values2);

}

}

return (det);

}

public static double[,] Inverse(int n, double[,] values)

{

int i, j, ii, jj, i1, j1;

double det;

double[,] values2 = new double[n - 1, n - 1];

double[,] values3 = new double[n, n];

for (j = 0; j < n; j++)

{

for (i = 0; i < n; i++)

{

i1 = 0;

for (ii = 0; ii < n; ii++)

{

if (ii == i)

continue;

j1 = 0;

for (jj = 0; jj < n; jj++)

{

if (jj == j)

continue;

values2[i1, j1] = values[ii, jj];

j1++;

}

i1++;

}

det = Determ(n - 1, values2);

values3[i, j] = Math.Pow(-1.0, i + j + 2.0) \* det;

}

}

det = Determ(n, values);

double temp;

for (i = 0; i < n; i++)

{

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

{

temp = values3[i, j];

values3[i, j] = values3[j, i];

values3[j, i] = temp;

}

}

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

values3[i, j] = values3[i, j] \* (1 / det);

}

}

return values3;

}

}

}

Form2:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Newtonsoft.Json;

using System.IO;

namespace Kursach2\_WF\_

{

public partial class Form2 : Form

{

private int opt;

private string number;

private string[,] values;

private string[,] svalues;

private string[,] values2;

private bool aim;

private bool loadall;

public Form2(int opt, string[,] values, string[,] values2, string[,] svalues, string number)

{

InitializeComponent();

this.opt = opt;

this.number = number;

this.values = values;

this.values2 = values2;

this.svalues = svalues;

if (values2 == null)

{

button3.Enabled = false;

}

}

public class files

{

public int option { get; set; }

public string number { get; set; }

public string[,] firstm { get; set; }

public string[,] secondm { get; set; }

public string[,] answer { get; set; }

}

private void button1\_Click(object sender, EventArgs e)

{

SaveFileDialog SaveFileDialog1 = new SaveFileDialog();

SaveFileDialog1.Filter = "Json Files (\*.json)|\*.json";

SaveFileDialog1.FileName = "default";

SaveFileDialog1.DefaultExt = "json";

var save = new files();

save.option = opt;

save.number = number;

save.firstm = values;

save.secondm = svalues;

save.answer = values2;

string output = JsonConvert.SerializeObject(save, Formatting.Indented);

if (SaveFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

using (StreamWriter sw = new StreamWriter(SaveFileDialog1.FileName))

sw.WriteLine(output);

}

catch

{

MessageBox.Show("Не удалось записать файл.\n Возможно он открыт в другой программе");

}

}

aim = false;

}

private void button2\_Click(object sender, EventArgs e)

{

SaveFileDialog SaveFileDialog1 = new SaveFileDialog();

SaveFileDialog1.Filter = "Json Files (\*.json)|\*.json";

SaveFileDialog1.FileName = "default";

SaveFileDialog1.DefaultExt = "json";

var save = new files();

save.option = opt;

save.number = number;

save.firstm = values;

save.secondm = svalues;

string output = JsonConvert.SerializeObject(save, Formatting.Indented);

if (SaveFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

using (StreamWriter sw = new StreamWriter(SaveFileDialog1.FileName))

sw.WriteLine(output);

}

catch

{

MessageBox.Show("Не удалось записать файл.\n Возможно он открыт в другой программе");

}

}

aim = false;

}

private void button3\_Click(object sender, EventArgs e)

{

SaveFileDialog SaveFileDialog1 = new SaveFileDialog();

SaveFileDialog1.Filter = "Json Files (\*.json)|\*.json";

SaveFileDialog1.FileName = "default";

SaveFileDialog1.DefaultExt = "json";

var save = new files();

save.option = opt;

save.answer = values2;

string output = JsonConvert.SerializeObject(save, Formatting.Indented);

if (SaveFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

using (StreamWriter sw = new StreamWriter(SaveFileDialog1.FileName))

sw.WriteLine(output);

}

catch

{

MessageBox.Show("Не удалось записать файл.\n Возможно он открыт в другой программе");

}

}

aim = false;

}

private void button4\_Click(object sender, EventArgs e)

{

OpenFileDialog OpenFileDialog1 = new OpenFileDialog();

OpenFileDialog1.Filter = "Json Files (\*.json)|\*.json";

OpenFileDialog1.FileName = "default.json";

OpenFileDialog1.DefaultExt = "json";

if (OpenFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

files open = JsonConvert.DeserializeObject<files>(File.ReadAllText(OpenFileDialog1.FileName));

opt = open.option;

number = open.number;

values = open.firstm;

svalues = open.secondm;

values2 = open.answer;

aim = true;

loadall = true;

}

catch

{

MessageBox.Show("Не удалось считать файл " + OpenFileDialog1.FileName + "\nВозможно файл поврежден");

aim = false;

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

OpenFileDialog OpenFileDialog1 = new OpenFileDialog();

OpenFileDialog1.Filter = "Json Files (\*.json)|\*.json";

OpenFileDialog1.FileName = "default.json";

OpenFileDialog1.DefaultExt = "json";

if (OpenFileDialog1.ShowDialog() == DialogResult.OK)

{

try

{

files open = JsonConvert.DeserializeObject<files>(File.ReadAllText(OpenFileDialog1.FileName));

opt = open.option;

number = open.number;

values = open.firstm;

svalues = open.secondm;

aim = true;

loadall = false;

}

catch

{

MessageBox.Show("Не удалось считать файл " + OpenFileDialog1.FileName + "\nВозможно файл поврежден");

aim = false;

}

}

}

public int retopt

{

get { return opt; }

}

public string retnumber

{

get { return number; }

}

public string[,] retvalues

{

get { return values; }

}

public string[,] retsvalues

{

get { return svalues; }

}

public string[,] retvalues2

{

get { return values2; }

}

public bool retaim

{

get { return aim; }

}

public bool retloadall

{

get { return loadall; }

}

}

}