Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа Ш2-2

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Инкапсуляция»

Выполнил: ст. гр. ТУУ-111

Михалин А.В

Вариант №2

12.06.2025

(дата выполнения)

Проверил: к.т.н., доц. Сафронов А.И.

(дата приёмки)

Москва – 2025 г.

**1. Цель**

Научиться **создавать инкапсулированный класс "Матрица" в режиме** разработки Windows Forms Application в интегрированной среде разработки *Microsoft Visual Studio на языке Visual C#.*

**2. Формулировка Задачи**

Создайте инкапсулированный класс "Матрица"

Подсказка: характерной особенностью инкапсулированного класса является отсутствие прямого доступа к его полям.

Предоставить решение (Solution), аналогичное рассмотренному на лекции для комплексных чисел.

Продумать расчёт собственных значений матриц, удобные методы, удобный ГПИ.

Сети Петри для ГПИ по заданию принципиальна**.**

**3.Сеть Петри**



Рисунок 3.1 – Сеть Петри

**4.Подбор тестовых примеров**

**Состояние 1:**

Программа запущенна и видны два поля ввода: строк (rows) и столбцов(cols) а также кнопка “создать матрицу” и поле для вывода матриц.

**Состояние 2:**

Пользователь вводит данные в поля и нажимает кнопку “создать матрицу”

**Состояние 3:**

Пользователь вводит некорректные данные и получает побудительное оповещение

**5.Листинг**

Cls.cs:

using System;

using System.Windows.Forms;

namespace MatrixApp

{

public partial class Cls : Form

{

// Конструктор главной формы

public Cls()

{

// Инициализация компонентов формы (кнопок, текстовых полей и т.д.)

InitializeComponent();

}

// Обработчик события нажатия кнопки "Создать матрицу"

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

{

// Парсим введенные пользователем значения размеров матрицы

int rows1, cols1;

if (int.TryParse(textBoxRows.Text, out rows1)& int.TryParse(textBoxRows.Text, out cols1))//проверка формата ввода колличества строк и столбцов

{

// Создаем экземпляр матрицы с указанными размерами

Matrix matrix = new Matrix(rows1, cols1);

// Заполняем матрицу примерными данными (значение = сумма индексов)

for (int i = 0; i < rows1; i++)

for (int j = 0; j < cols1; j++)

matrix.SetElement(i, j, i + j); // Установка значения в позицию [i,j]

// Выводим текстовое представление матрицы в интерфейс

textBoxOutput.Text = matrix.ToString();

}

else

{

// Обработка ошибок с выводом сообщения пользователю

MessageBox.Show("Ошибка: Вводная строка имела не верный формат! введите целые числа");

}

}

}

}

class Matrix:

using System;

namespace MatrixApp

{

public class Matrix

{

// Двумерный массив для хранения элементов матрицы

private double[,] data;

// Свойство для получения количества строк (только чтение)

public int Rows { get; private set; }

// Свойство для получения количества столбцов (только чтение)

public int Columns { get; private set; }

// Конструктор матрицы

public Matrix(int rows, int columns)

{

// Проверка корректности размеров матрицы

if (rows <= 0 || columns <= 0)

throw new ArgumentException("Размеры матрицы должны быть положительными.");

// Установка размеров матрицы

Rows = rows;

Columns = columns;

// Выделение памяти под двумерный массив элементов

data = new double[rows, columns];

}

// Метод установки значения элемента

public void SetElement(int row, int column, double value)

{

// Проверка выхода индекса за границы массива

if (row < 0 || row >= Rows || column < 0 || column >= Columns)

throw new IndexOutOfRangeException("Индекс вне границ массива.");

// Присвоение значения ячейке матрицы

data[row, column] = value;

}

// Метод получения значения элемента

public double GetElement(int row, int column)

{

// Проверка выхода индекса за границы массива

if (row < 0 || row >= Rows || column < 0 || column >= Columns)

throw new IndexOutOfRangeException("Индекс вне границ массива.");

// Возврат значения из указанной ячейки

return data[row, column];

}

// Метод преобразования матрицы в строку

public override string ToString()

{

// Инициализация пустой строки для результата

string result = "";

// Цикл по строкам матрицы

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

{

// Цикл по столбцам матрицы

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

{

// Форматирование числа (2 знака после запятой) и добавление табуляции

result += data[i, j].ToString("F2") + "\t";

}

// Добавление перевода строки после окончания строки матрицы

result += Environment.NewLine;

}

// Возврат итоговой строки

return result;

}

}

}

**6.Расчет тестовых примеров**

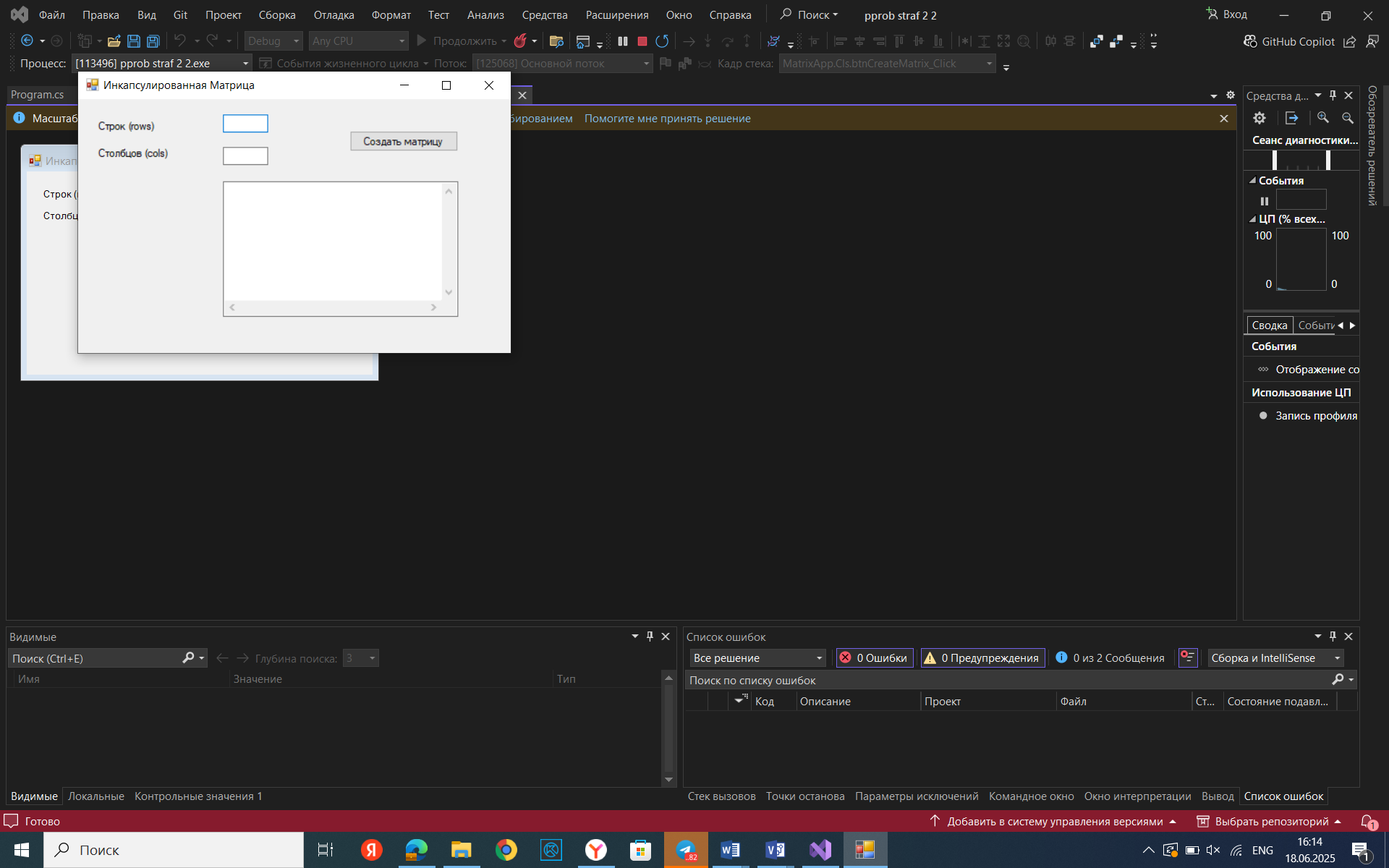
****

Рисунок 6.1 – Состояние №1

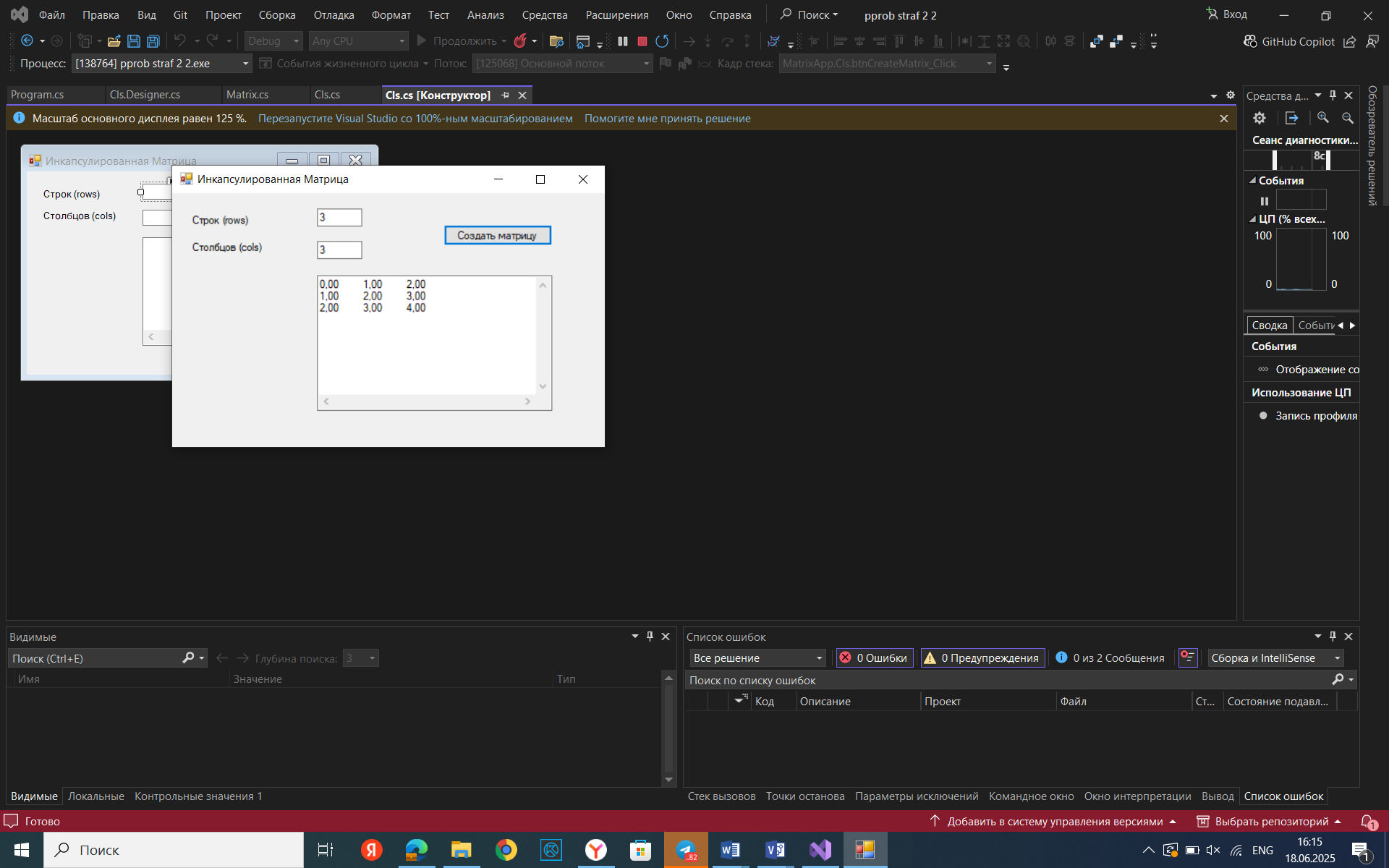
****

Рисунок 6.2 – Состояние №2

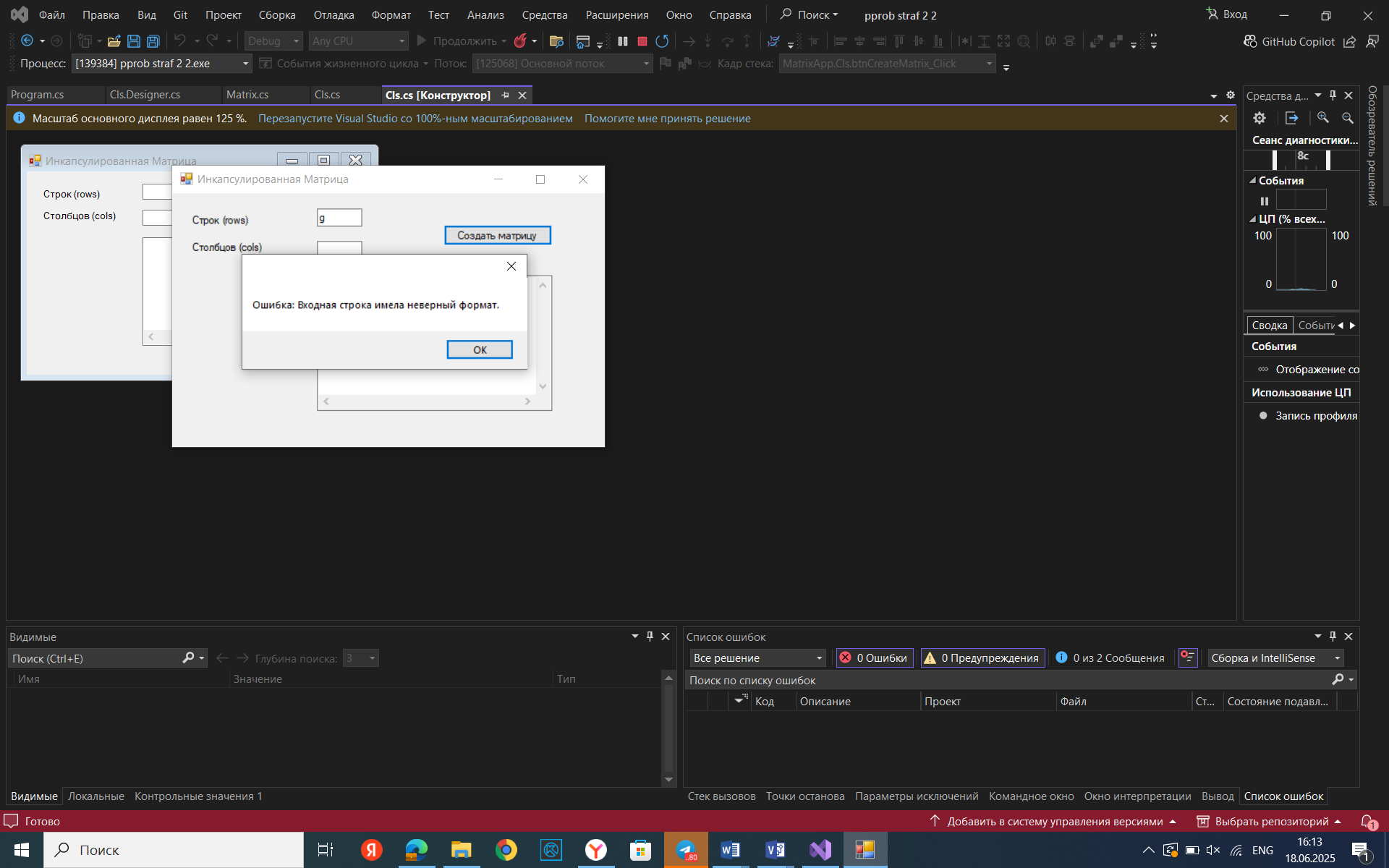
****

Рисунок 6.3 – Состояние №3

**7.Вывод**

В ходе выполнения лабораторной работы была разработана учебная Windows Forms-программа с графическим пользовательским интерфейсом, демонстрирующая основы работы c инкапсулированными матрицами управления в среде Windows Forms. Были получены навыки в **создании инкапсулированного класса в режиме** разработки Windows Forms Application в интегрированной среде разработки *Microsoft Visual Studio на языке Visual C#.*