

Лабораторная работа №4

Тема: Знакомство технологией создания приложений WPF и языком разметки XAML.

Цель работы: Формирование базовых знаний по данной теме.

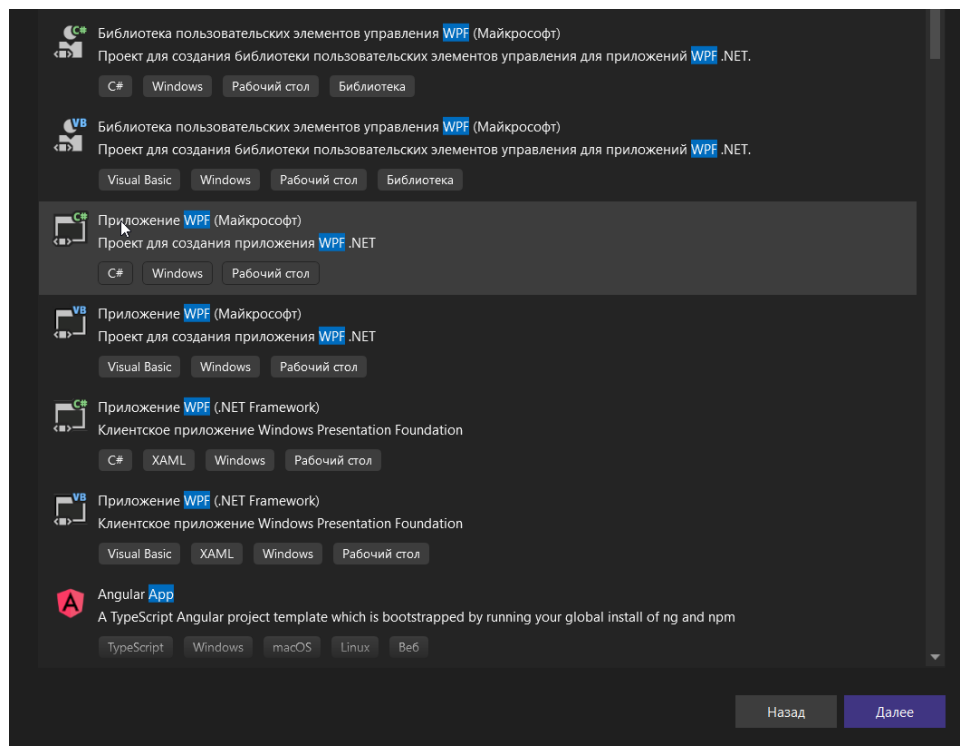
Получение навыков работы по созданию приложений при помощи технологии WPF в Visual Studio. Научится добавлять компоненты на форму, используя язык разметки XAML.

Ход работы

Вариант 8

Задание 1: Разработать приложение с помощью технологии WPF и языка разметки XAML. Создать двумерный массив и вывести его по нажатию на командную кнопку на форме.

Дана квадратная матрица A порядка n . Составить программу, которая находит минимальный элемент и все элементы, расположенные в одной строке и в одном столбце с минимальным заменяет на минимальное значение. Учесть, что значения в массиве не повторяются.



Настроить новый проект

Приложение WPF (Майкрософт)

C#

Windows

Рабочий стол

Имя проекта

Семестр2_ЛР4_Задание_1

Расположение

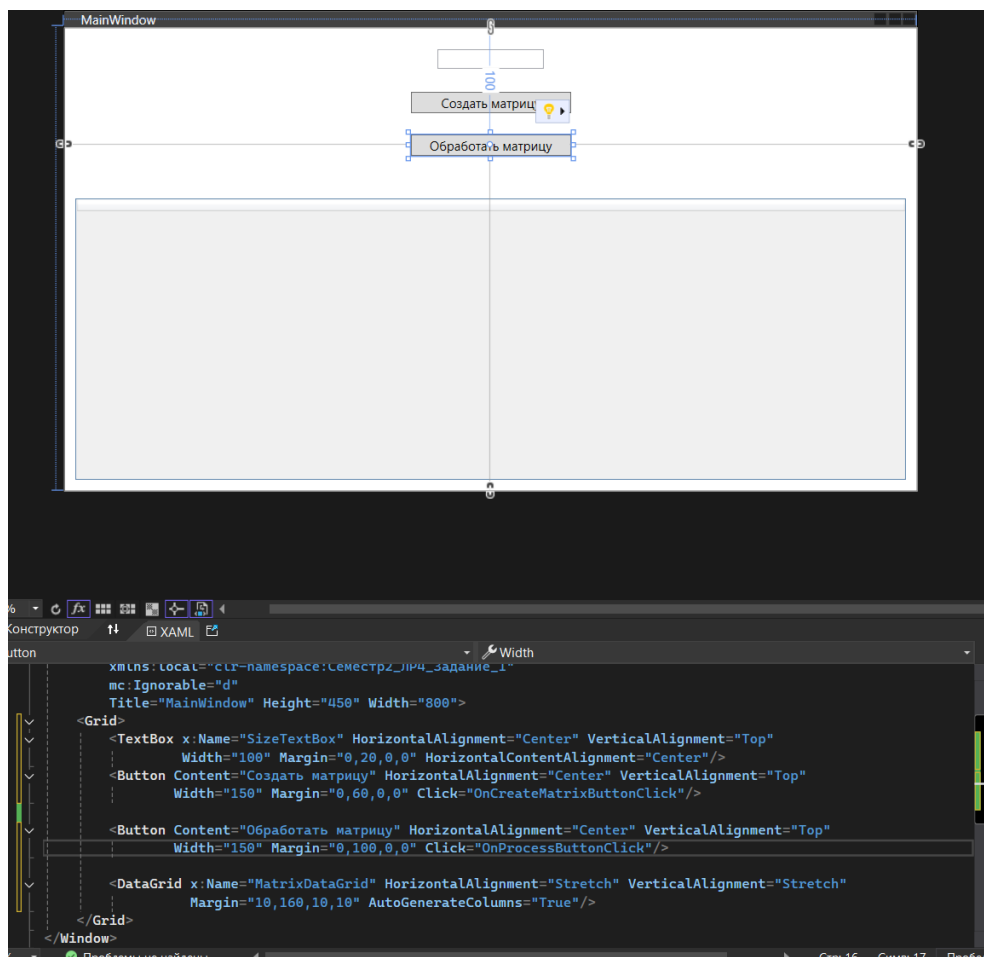
C:\Users\Mikhail\Documents\repos\University-Homework\Information_Technology\Семестр2_ЛР4\

Имя решения

Семестр2_ЛР4_Задание_1

☐ Поместить решение и проект в одном каталоге

Проект будет создан в "C:\Users\Mikhail\Documents\repos\University-Homework\Information_Technology\Семестр2_ЛР4\Семестр2_ЛР4_Задание_1\Семестр2_ЛР4_Задание_1\"



Код:

```
1 using System.Data;
2 using System.Text;
3 using System.Windows;
4 using System.Windows.Controls;
5 using System.Windows.Data;
6 using System.Windows.Documents;
7 using System.Windows.Input;
8 using System.Windows.Media;
9 using System.Windows.Media.Imaging;
10 using System.Windows.Navigation;
11 using System.Windows.Shapes;
```

```
namespace Семестр2_ЛР4_Задание_1
{
    Ссылка: 2
    public partial class MainWindow : Window
    {
        private int[,] matrix;
        private int size;

        Ссылка: 0
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

```
// Нажмите кнопки для создания матрицы
Ссылка: 1
private void OnCreateMatrixButtonClick(object sender, RoutedEventArgs e)
{
    // Проверяем, что пользователь ввел размер
    if (int.TryParse(SizeTextBox.Text, out size) && size > 0)
    {
        matrix = new int[size, size];

        // Заполняем матрицу случайными числами
        Random random = new Random();
        for (int i = 0; i < size; i++)
        {
            for (int j = 0; j < size; j++)
            {
                matrix[i, j] = random.Next(1, 100);
            }
        }

        MatrixDataGrid.ItemsSource = ConvertMatrixToDataTable(matrix).DefaultView;
    }
    else
    {
        MessageBox.Show("Пожалуйста, введите размер матрицы.", "Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

```
// Нажатия кнопки для обработки матрицы
Ссылка: 1
private void OnProcessButtonClick(object sender, RoutedEventArgs e)
{
    if (matrix == null)
    {
        MessageBox.Show("Сначала создайте матрицу.", "Ошибка", MessageBoxButton.OK,
            MessageBoxImage.Warning);
        return;
    }

    // Находим минимальный элемент в матрице
    int minValue = matrix[0, 0];
    int minRow = 0, minCol = 0;

    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (matrix[i, j] < minValue)
            {
                minValue = matrix[i, j];
                minRow = i;
                minCol = j;
            }
        }
    }

    // Заменяем все элементы в строке и столбце на минимальное значение
    for (int i = 0; i < size; i++)
    {
        matrix[minRow, i] = minValue;
        matrix[i, minCol] = minValue;
    }

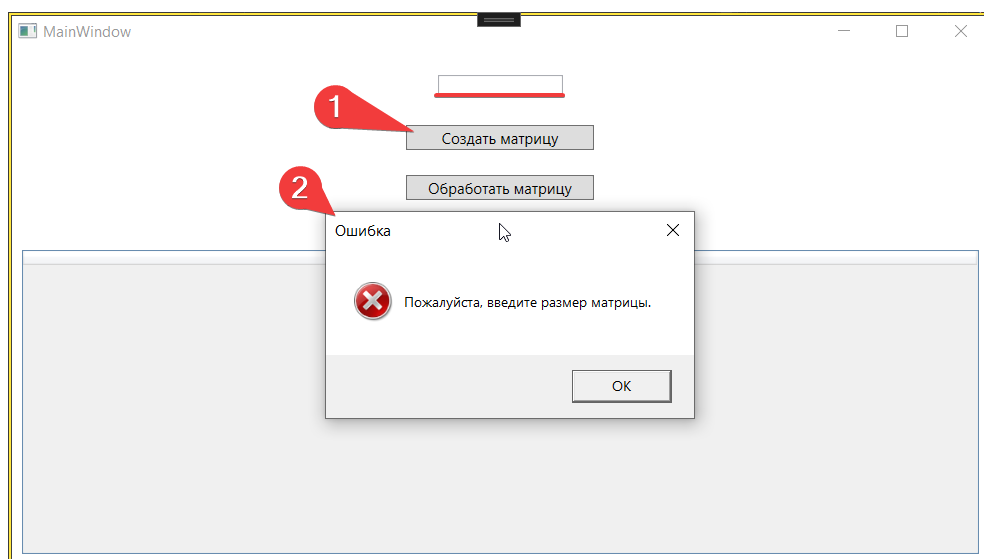
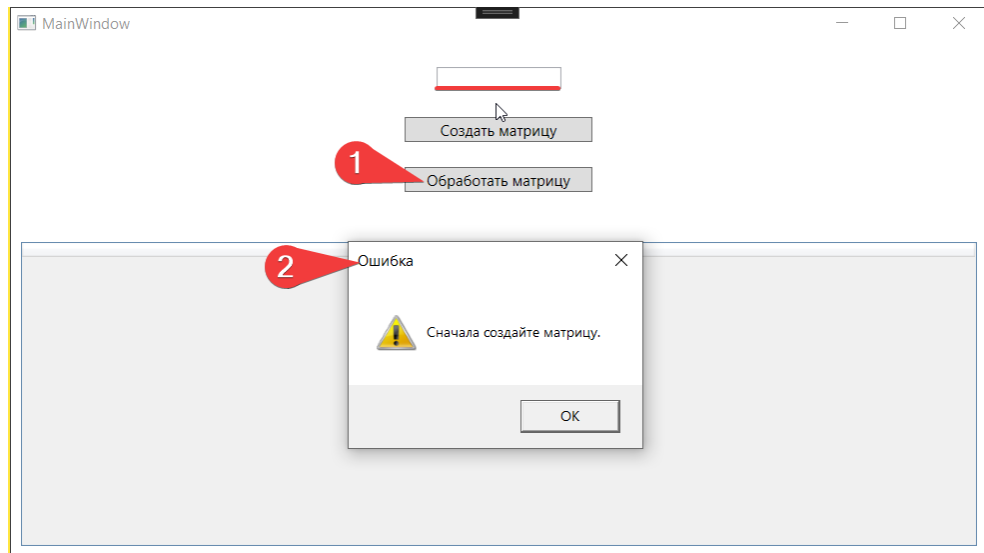
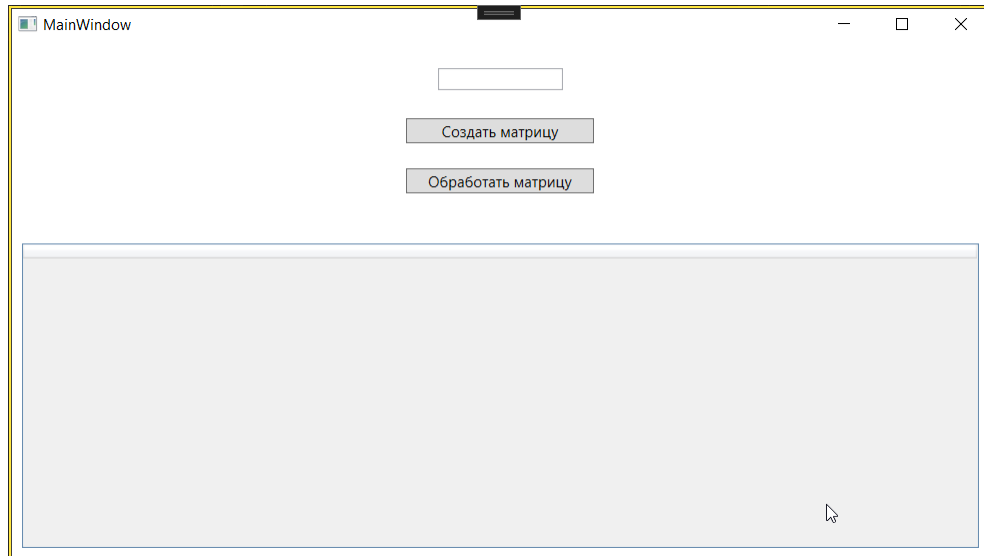
    // Обновляем отображение матрицы в DataGrid
    MatrixDataGrid.ItemsSource = ConvertMatrixToDataTable(matrix).DefaultView;
}
}
```

```
// Преобразование матрицы в DataTable
Ссылка: 2
private DataTable ConvertMatrixToDataTable(int[,] matrix)
{
    var table = new DataTable();
    for (int i = 0; i < size; i++)
    {
        table.Columns.Add($"Column{i + 1}");
    }

    for (int i = 0; i < size; i++)
    {
        var row = table.NewRow();
        for (int j = 0; j < size; j++)
        {
            row[j] = matrix[i, j];
        }
        table.Rows.Add(row);
    }

    return table;
}
}
```

Тестирование:



MainWindow

8

Создать матрицу

Обработать матрицу

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
74	14	24	68	74	30	21	25
14	81	74	92	66	17	73	96
91	7	53	36	38	98	13	52
42	30	91	35	49	24	78	8
64	51	56	58	11	29	10	40
46	67	10	54	26	70	33	65
34	38	83	69	78	12	41	88
4	49	3	77	7	90	85	21

MainWindow

8

Создать матрицу

Обработать матрицу

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
74	14	3	68	74	30	21	25
14	81	3	92	66	17	73	96
91	7	3	36	38	98	13	52
42	30	3	35	49	24	78	8
64	51	3	58	11	29	10	40
46	67	3	54	26	70	33	65
34	38	3	69	78	12	41	88
3	3	3	3	3	3	3	3

Обработка матрицы

*

после изменил
название формы*

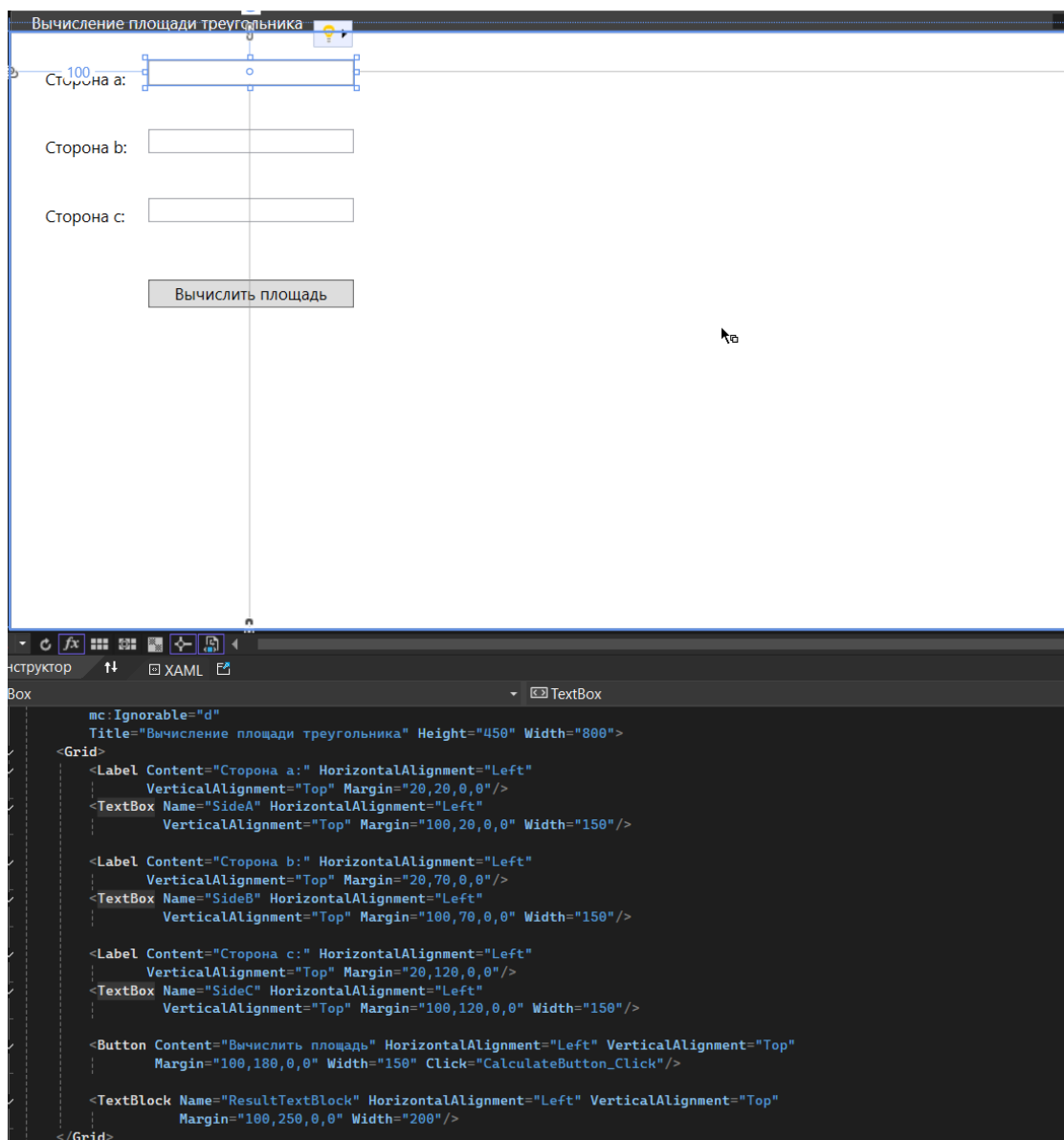
Задание 2: Разработать приложение с помощью технологии WPF и языка разметки XAML. Необходимое количество полей для ввода создается

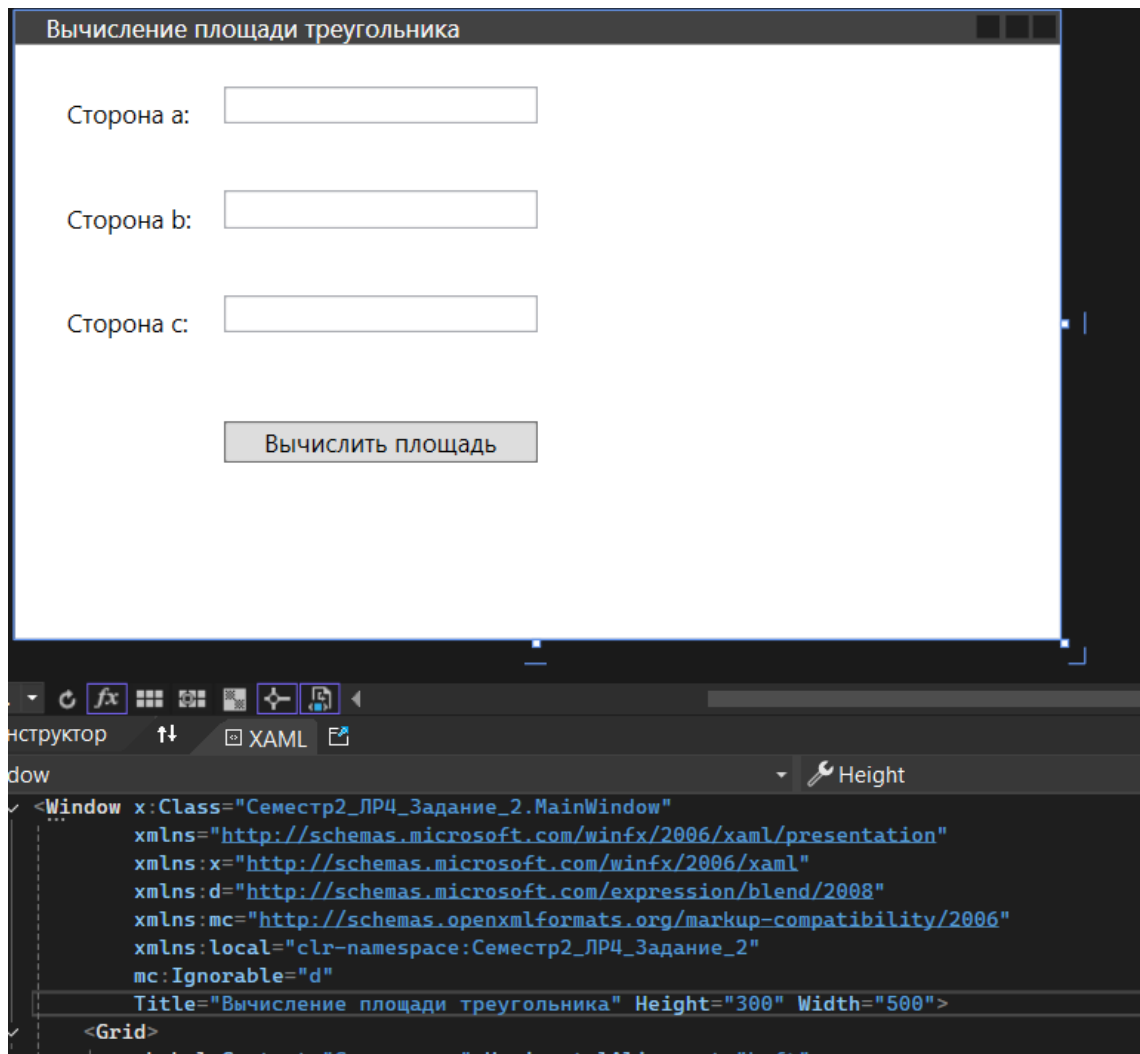
индивидуально для каждого варианта. Результат вычислений выводится в поле для вывода.

Вычисление площади треугольника по формуле Герона. S – площадь, p – полупериметр, a, b, c – стороны треугольника. Стороны треугольника вводятся в поля для ввода, а площадь треугольника выводится в окно вывода.

$$p = \frac{a + b + c}{2}$$

$$S = \sqrt{p * (p - a) * (p - b) * (p - c)}$$





Код:

```
Ссылка: 0
private void CalculateButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        double a = Convert.ToDouble(SideA.Text);
        double b = Convert.ToDouble(SideB.Text);
        double c = Convert.ToDouble(SideC.Text);

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

        // Площадь
        double area = Math.Sqrt(p * (p - a) * (p - b) * (p - c));

        ResultTextBlock.Text = $"Площадь: {area:F2}";
    }
    catch (Exception ex)
    {
        ResultTextBlock.Text = "Некорректный ввод!";
    }
}
```


Тестирование:

Вычисление площади треугольника

Сторона a:

Сторона b:

Сторона c:

Вычислить площадь

Вычисление площади треугольника

Сторона a:

Сторона b:

Сторона c:

Вычислить площадь

Некорректный ввод!

Вычисление площади треугольника

Сторона a:

Сторона b:

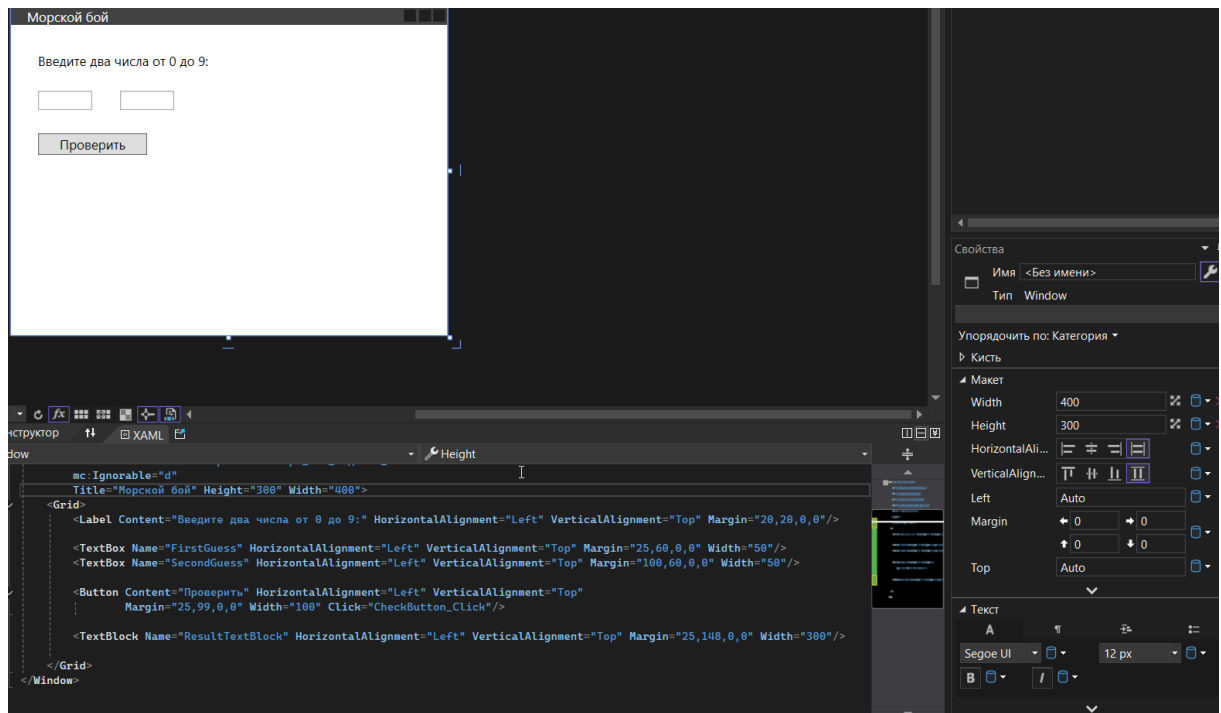
Сторона c:

Вычислить площадь

Площадь: 10,39

Задание 3: Разработать приложение с помощью технологии WPF и языка разметки XAML. Программирование разветвляющегося алгоритма.

Морской бой. Машина задумывает два числа от 0 до 9. Игрок пытается их угадать, вводя свои два числа. Если они совпали (в любом сочетании), то игрок выиграл.



Код:

```
using System.Threading.Tasks;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Семестр2_ЛР4_Задание_3
{
    Ссылка: 2
    public partial class MainWindow : Window
    {
        private int num1;
        private int num2;

        Ссылка: 0
        public MainWindow()
        {
            InitializeComponent();
            Random rand = new Random();
            num1 = rand.Next(3, 5);
            num2 = rand.Next(3, 5);
        }
    }
}
```

```

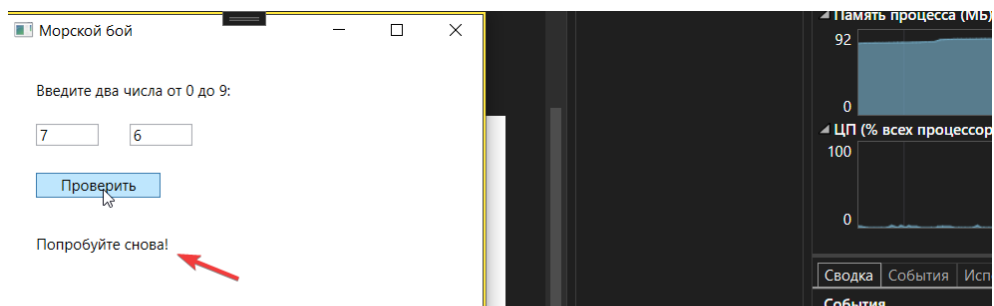
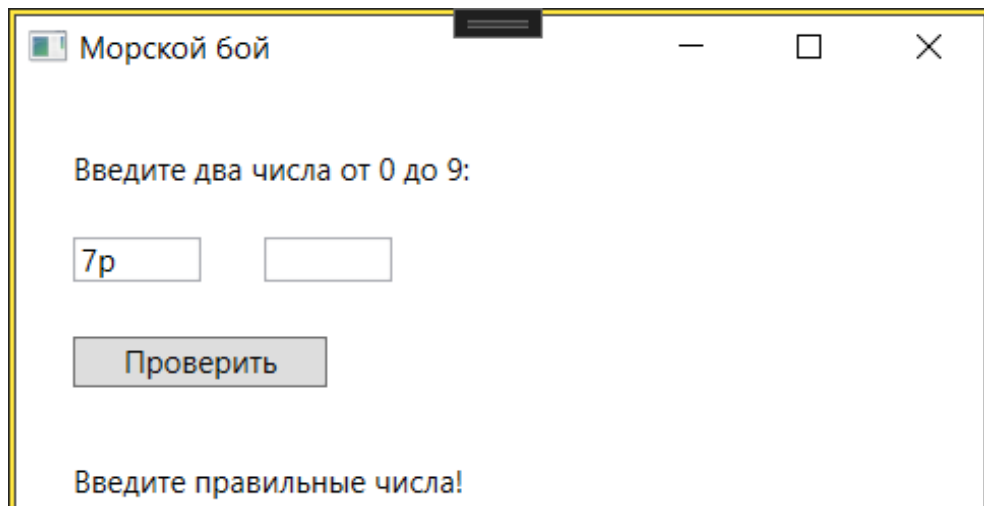
28  private async void CheckButton_Click(object sender, RoutedEventArgs e)
29  {
30      try
31      {
32          int guess1 = Convert.ToInt32(FirstGuess.Text);
33          int guess2 = Convert.ToInt32(SecondGuess.Text);
34
35          // Проверяем, что введенные числа в пределах допустимого диапазона (0-9)
36          if (guess1 < 0 || guess1 > 9 || guess2 < 0 || guess2 > 9)
37          {
38              ResultTextBlock.Text = "Числа должны быть от 0 до 9!";
39              return;
40          }
41
42          // Проверка, совпали ли числа
43          if ((guess1 == num1 && guess2 == num2) || (guess1 == num2 && guess2 == num1))
44          {
45              ResultTextBlock.Text = "Вы выиграли!";
46          }
47          else
48          {
49              ResultTextBlock.Text = "Попробуйте снова!";
50
51              // Задержка на 1 секунду, затем очищаем для интерактивности
52              await Task.Delay(1000);
53              ResultTextBlock.Text = "";
54          }
55      }
56      catch (FormatException)
57      {
58          ResultTextBlock.Text = "Введите правильные числа!";
59      }
60  }
61
62  }

```

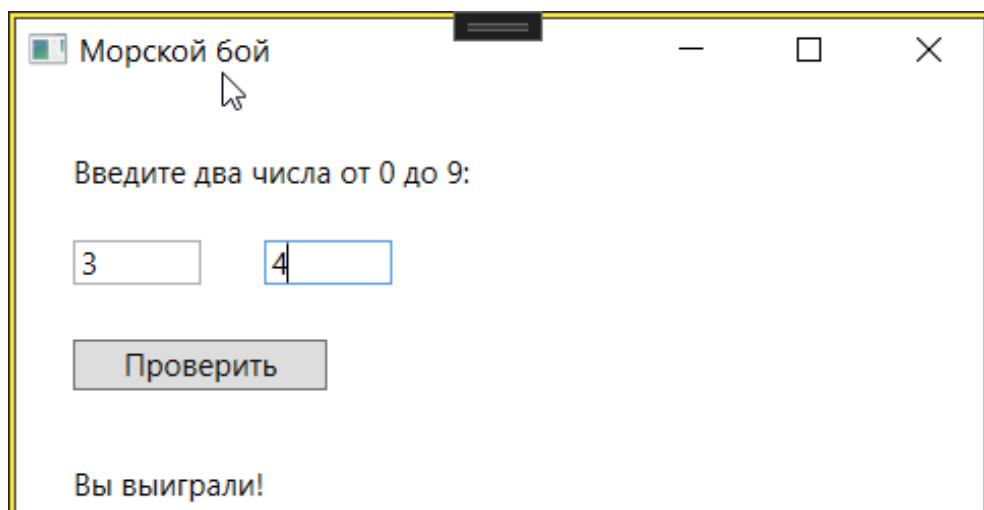
Тестирование:

Морской бой

Введите два числа от 0 до 9:

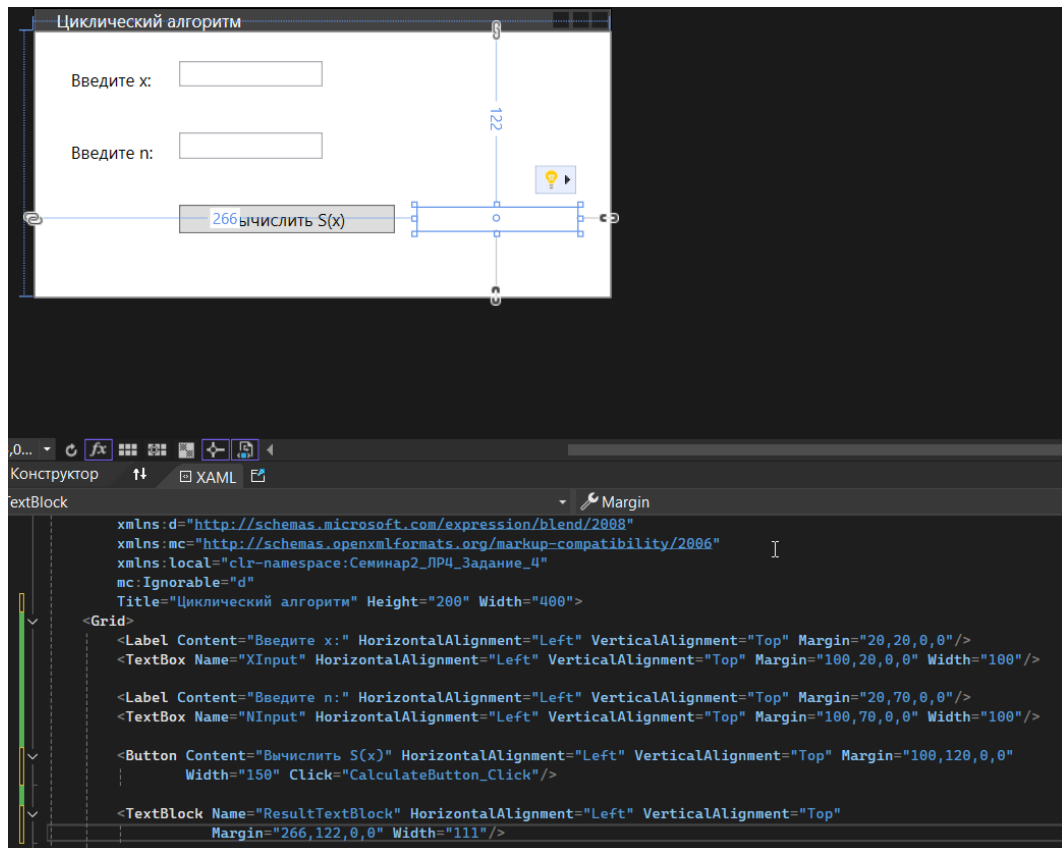


Надпись "Попробуйте
снова" появляется лишь на
секунду для
интерактивности



Задание 4: Разработать приложение с помощью технологии WPF и языка разметки XAML. Программирование циклического алгоритма.

$$8. S(x) = \prod_{k=0}^n \left(1 + \frac{\sin(kx)}{(4x-k)}\right)$$



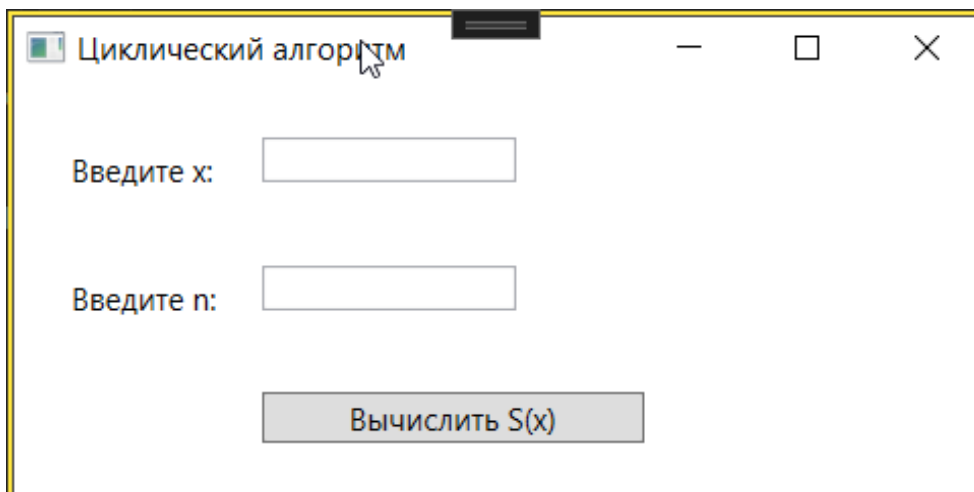
Код:

```
private void CalculateButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        double x = Convert.ToDouble(XInput.Text);
        int n = Convert.ToInt32(NInput.Text);

        double result = 1;
        for (int k = 0; k <= n; k++)
        {
            result *= (1 + Math.Sin(k * x) / (4 * x - k));
        }

        ResultTextBlock.Text = $"S(x) = {result}";
    }
    catch (Exception ex)
    {
        ResultTextBlock.Text = "Ошибка ввода!";
    }
}
```

Тестирование:

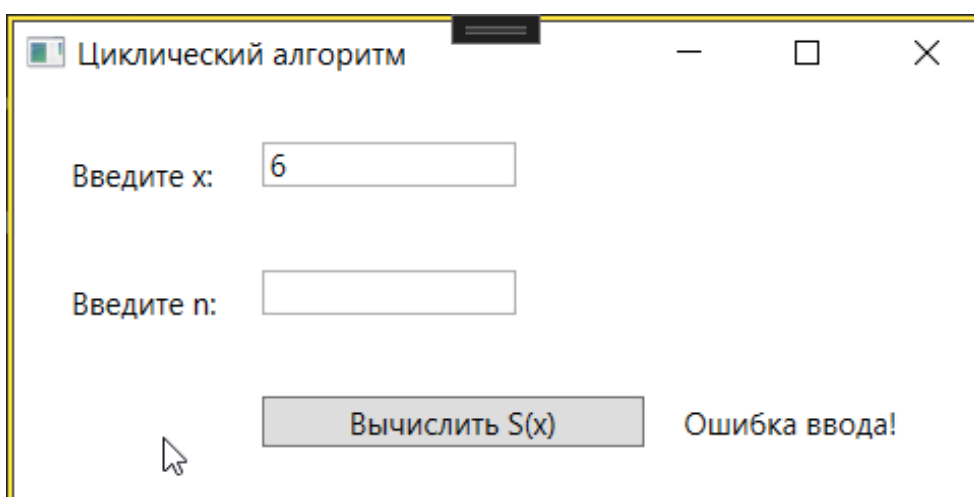


Циклический алгоритм

Введите x:

Введите n:

Вычислить S(x)

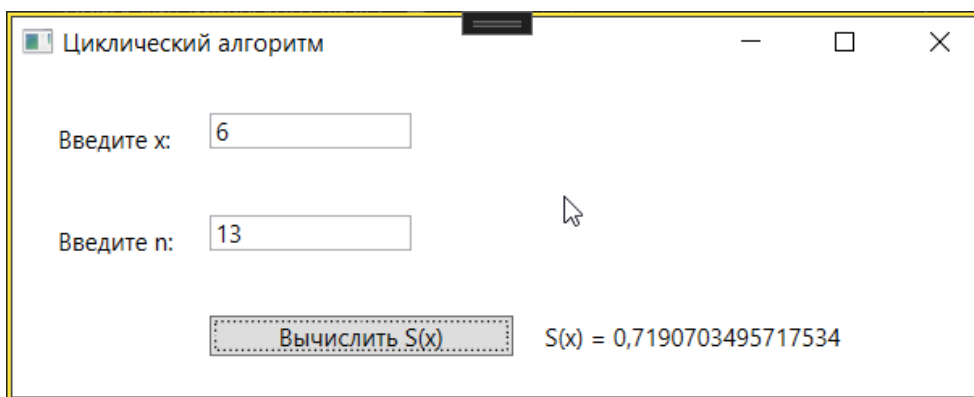


Циклический алгоритм

Введите x:

Введите n:

Вычислить S(x) Ошибка ввода!



Циклический алгоритм

Введите x:

Введите n:

Вычислить S(x) S(x) = 0,7190703495717534

Вывод: изучили базовые принципы работы с WPF в Visual Studio.

Научились добавлять компоненты на форму, используя язык разметки XAML, производить вычисления с циклическими операторами, условными, работать с матрицами.