**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ИВАНОВСКОЙ ОБЛАСТИ**

Областное государственное бюджетное

профессиональное образовательное учреждения

«Ивановский промышленно-экономический колледж»

**ОТЧЕТ**

**практической подготовки по учебной практике**

**УП.01.01.01** МДК.01.01. Разработка программных модулей

База практики:

**ОГБПОУ «Ивановский промышленно-экономический колледж»**

Сроки прохождения практической подготовки: **29.05.2023-03.06.2023**

Специальность: **09.02.07 Информационные системы и программирование**

Курс 2 Группа № **205**

Обучающийся Худобородова Валерия Дмитриевна

Фамилия, Имя, Отчество студента (*полностью*)

Руководитель

от Организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Доманская А.К.

оценка  подпись Ф.И.О.

г. Иваново

Содержание

[Введение 3](#_Toc137630124)

[Техника безопасности при работе за компьютером 4](#_Toc137630125)

[Практическая работа №1 6](#_Toc137630126)

[Практическая работа №2 13](#_Toc137630127)

[Практическая работа №3. 16](#_Toc137630128)

[Практическая работа №4 19](#_Toc137630129)

[Практическая работа №5 25](#_Toc137630130)

[Практическая работа №6 34](#_Toc137630131)

[Практическая работа №7 35](#_Toc137630132)

[Практическая работа №8 36](#_Toc137630133)

[Заключение 37](#_Toc137630134)

# Введение

# Техника безопасности при работе за компьютером

Поскольку персональный компьютер обладает всеми свойствами электрического прибора, то на него распространяются основные правила безопасности при взаимодействии с проводниками тока:

1. Нельзя размещать какие-либо вещи на проводах, а также самостоятельно менять их расположение без особой нужды;
2. Нельзя работать на ПК с мокрыми руками;
3. Нельзя очищать поверхность компьютера от загрязнений, когда он находится во включенном состоянии;
4. Недопустимо снимать корпус любой из составных частей ПК во время его работы;
5. Во время работы на компьютере нельзя одновременно прикасаться к другим металлическим конструкциям, которые стоят на той же поверхности. Это касается отопительных батарей или трубопроводов;
6. В помещении с компьютерами непозволительно курить или употреблять пищу непосредственно на рабочем месте;
7. При ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к ответственному за обслуживание компьютерной техники;
8. Расстояние между глазами пользователя и экраном составляет не менее полуметра. Но пользователь должен быть в состоянии дотянуться кончиками пальцев до верхнего края монитора;
9. Клавиатура размещается за 20-30 сантиметров от края стола;
10. Стул стоит таким образом, чтобы спина лишь немного упиралась в его спинку. Высота сидения позволяет держать ровную осанку;
11. Локти согнуты под прямым углом, а в кистях рук, лежащих на столе, не чувствуется напряжения;
12. Локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки;
13. Вокруг стола не должно быть свисающих проводов, пользователь не должен контактировать с ними;

Практическая работа №1

Работа с массивами данных, вызов консоли.

Цель работы: Научиться создавать массивы и программы в Visual Studio.

Задание 1:

Создать двумерный массив размерностью m на n элементов, заполнив его случайными числами в диапазоне от 0 до 9. Отсортировать элементы массива по возрастанию вначале по строкам, а затем по столбцам. Вывести на экран исходный массив, массив, отсортированный построчно, массив, отсортированный по столбцам. Для сортировки применить метод пузырька.

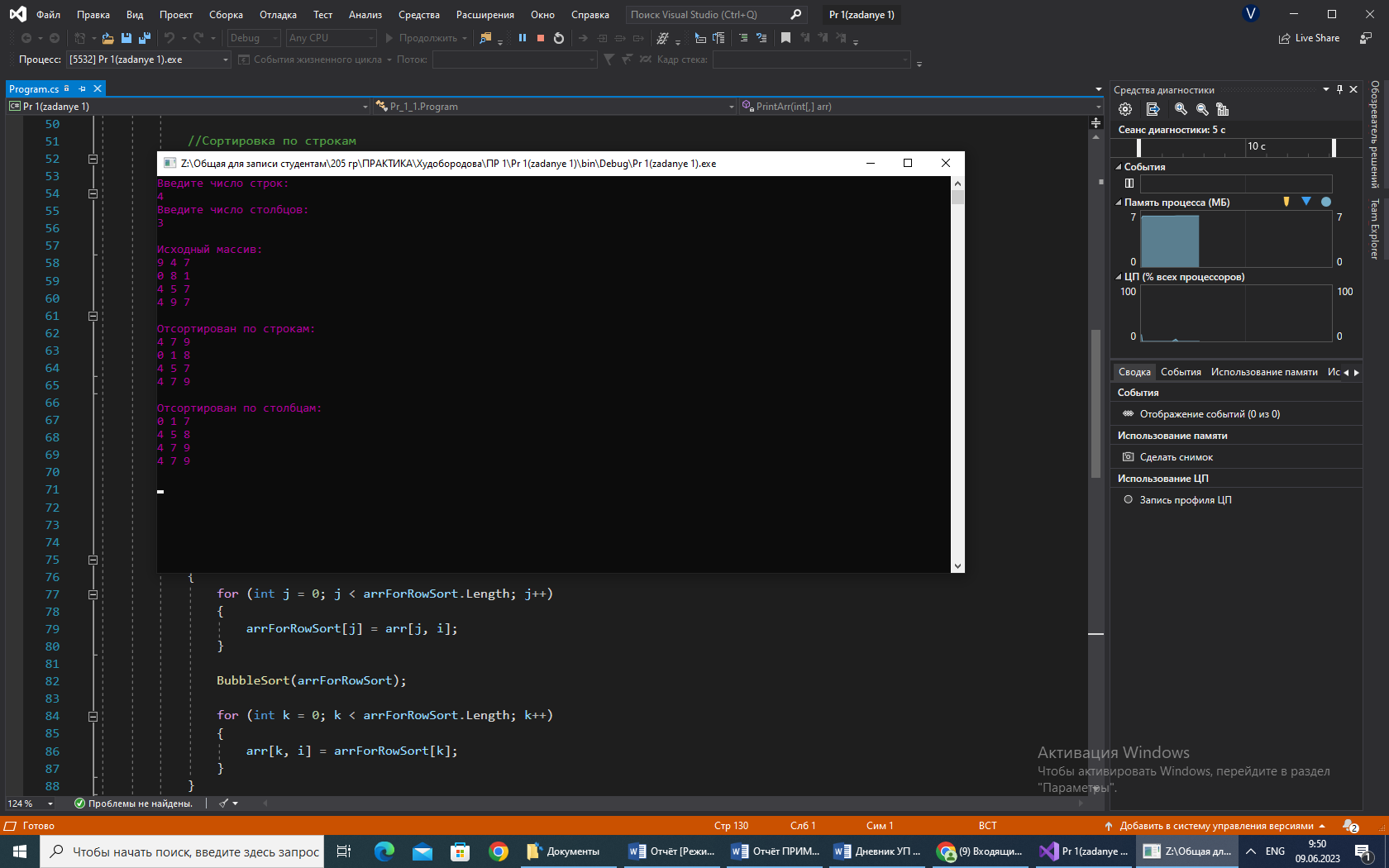


Рис.1 – Результат выполнения программы

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Pr\_1\_1

{

internal class Program

{

static void Main(string[] args)

{

Console.ForegroundColor = ConsoleColor.Magenta;

//Получаем размерность массива от пользователя

Console.WriteLine("Введите число строк: ");

int m = int.Parse(Console.ReadLine());

Console.WriteLine("Введите число столбцов: ");

int n = int.Parse(Console.ReadLine());

Console.WriteLine(); //Для красоты

//Создаём масив

int[,] arr = new int[m, n];

//Создаём рандом

Random rnd = new Random();

//Наполняем массив

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

{

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

{

arr[i, j] = rnd.Next(0, 10);

}

}

//Выводим массив

Console.WriteLine("Исходный массив:");

PrintArr(arr);

Console.WriteLine(); //Для красоты

//Массив для сортировки

int[] arrForStrSort = new int[n];

int[] arrForRowSort = new int[m];

//Сортировка по строкам

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

{

for (int k = 0; k < arrForStrSort.Length; k++)

{

arrForStrSort[k] = arr[i, k];

}

BubbleSort(arrForStrSort);

for (int d = 0; d < arrForStrSort.Length; d++)

{

arr[i, d] = arrForStrSort[d];

}

}

//Выводим отсортированный по строкам

Console.WriteLine("Отсортирован по строкам:");

PrintArr(arr);

Console.WriteLine(); //Для красоты

//Сортировка по столбцам

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

{

for (int j = 0; j < arrForRowSort.Length; j++)

{

arrForRowSort[j] = arr[j, i];

}

BubbleSort(arrForRowSort);

for (int k = 0; k < arrForRowSort.Length; k++)

{

arr[k, i] = arrForRowSort[k];

}

}

//Выводим отсортированный по столбцам

Console.WriteLine("Отсортирован по столбцам:");

PrintArr(arr);

Console.WriteLine(); //Для красоты

Console.ReadKey();

}

static void BubbleSort(int[] arr)

{

for (int i = 0; i < arr.Length; i++)

{

for (int j = 0; j < arr.Length - i - 1; j++)

{

if (arr[j] > arr[j + 1])

{

int num = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = num;

}

}

}

}

static void PrintArr(int[,] arr)

{

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

{

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

{

Console.Write(arr[i, j] + " ");

}

Console.WriteLine();

}

}

}

}

Задание 2:

Создать двумерный массив размерности 3x3, и заполнить его числами от 1 до 9, расположенными случайным образом.

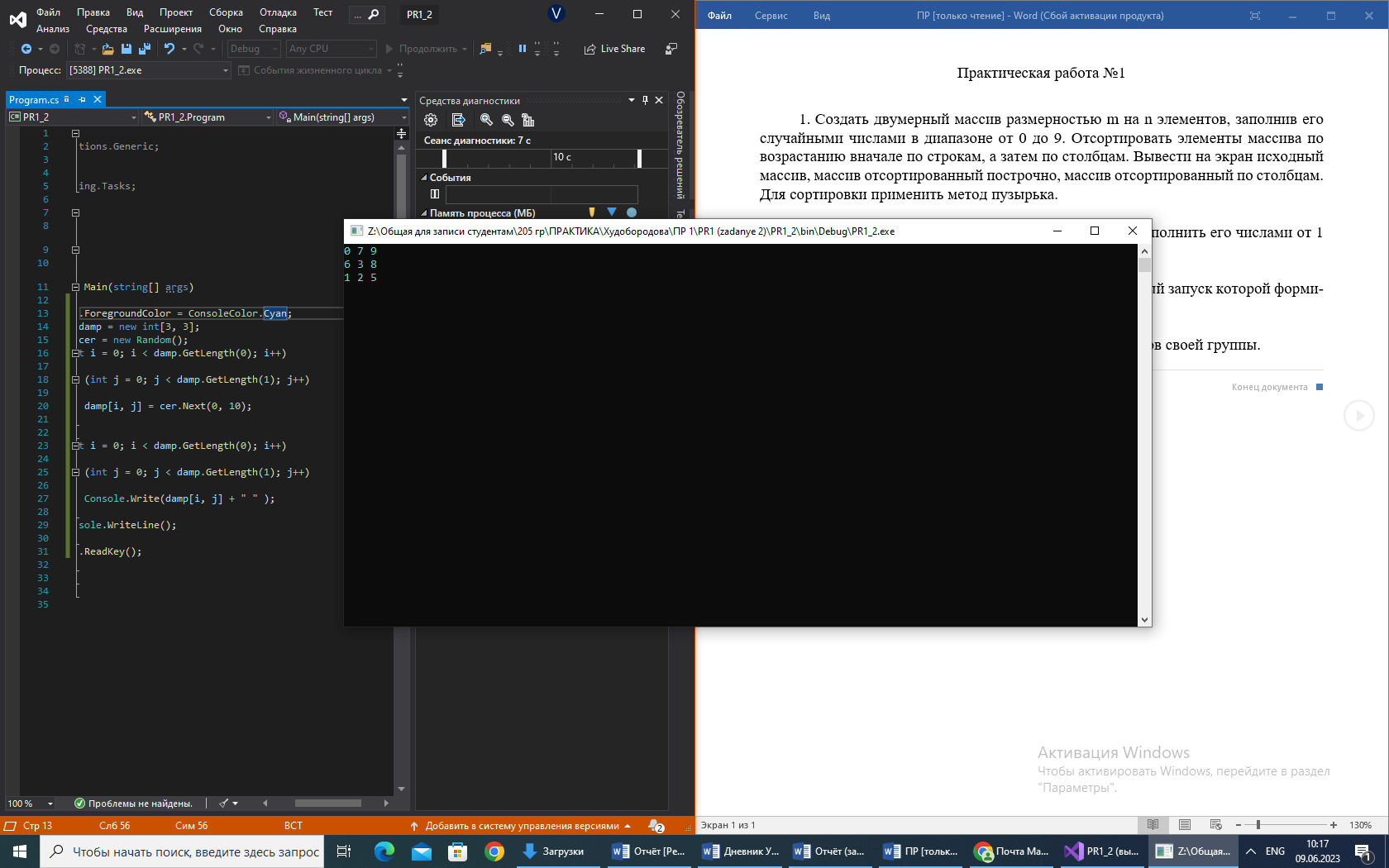


Рис. 2- Результат выполнения программы

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PR1\_2

{

class Program

{

static void Main(string[] args)

{

Console.ForegroundColor = ConsoleColor.;

int[,] damp = new int[3, 3];

Random cer = new Random();

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

{

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

{

damp[i, j] = cer.Next(0, 10);

}

}

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

{

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

{

Console.Write(damp[i, j] + " " );

}

Console.WriteLine();

}

Console.ReadKey();

}

}

}

Задание 3:

Создать программу конструктора бургеров, каждый запуск которой формирует, случайный рецепт из возможных ингредиентов.

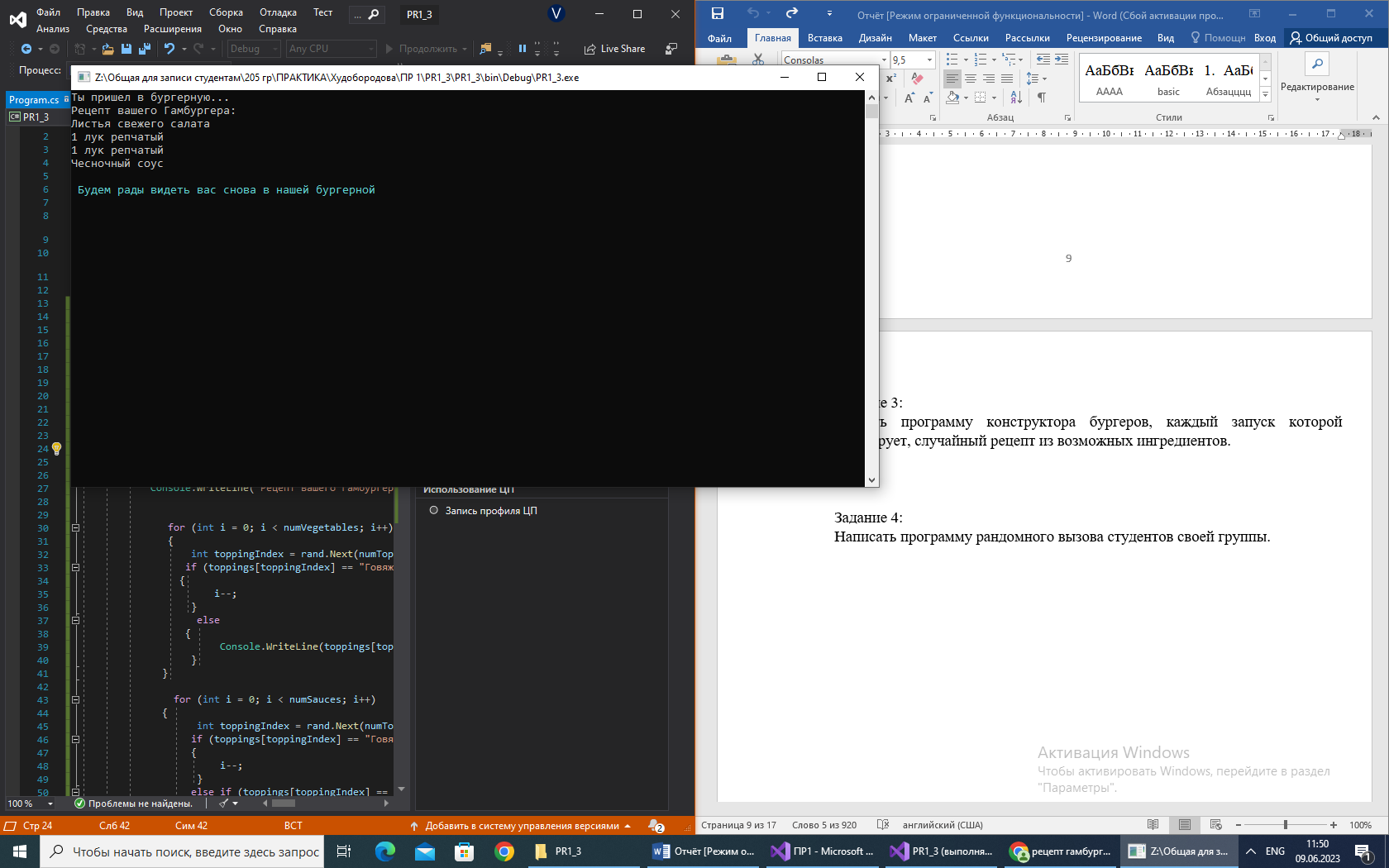


Рис.3 - Результат выполнения программы

Код программы:

static void Main(string[] args)

{

string[] toppings = { "Говяжья котлета", "Куринная котлета", "Бекон", "Сыр", "Листья свежего салата", "Помидор", "Маринованные огурцы", "Кетчуп", "Чесночный соус", "Сырный соус", "1 лук репчатый", "Булочка" };

int numToppings = toppings.Length;

Console.WriteLine("Ты пришел в бургерную...");

// Формирование случайного рецепта

Random rand = new Random();

int numPatties = rand.Next(3, 10);

int numVegetables = rand.Next(2, 5);

int numSauces = rand.Next(1, 3);

int newBylka = rand.Next(4, 6);

// Вывод рецепта на экран

Console.WriteLine("Рецепт вашего Гамбургера:");

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

{

int toppingIndex = rand.Next(numToppings);

if (toppings[toppingIndex] == "Говяжья котлета" || toppings[toppingIndex] == "Куринная котлета")

{

i--;

}

else

{

Console.WriteLine(toppings[toppingIndex]);

}

}

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

{

int toppingIndex = rand.Next(numToppings);

if (toppings[toppingIndex] == "Говяжья котлета" || toppings[toppingIndex] == "Куринная котлета")

{

i--;

}

else if (toppings[toppingIndex] == "Кетчуп" || toppings[toppingIndex] == "Чесночный соус" || toppings[toppingIndex] == "Сырный соус")

{

if (i > 0)

{

i--;

}

else

{

Console.WriteLine(toppings[toppingIndex]);

}

}

else

{

Console.WriteLine(toppings[toppingIndex]);

}

}

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("\n Будем рады видеть вас снова в нашей бургерной");

Console.ReadKey();

}

}

}

Задание 4:

Написать программу рандомного вызова студентов своей группы.

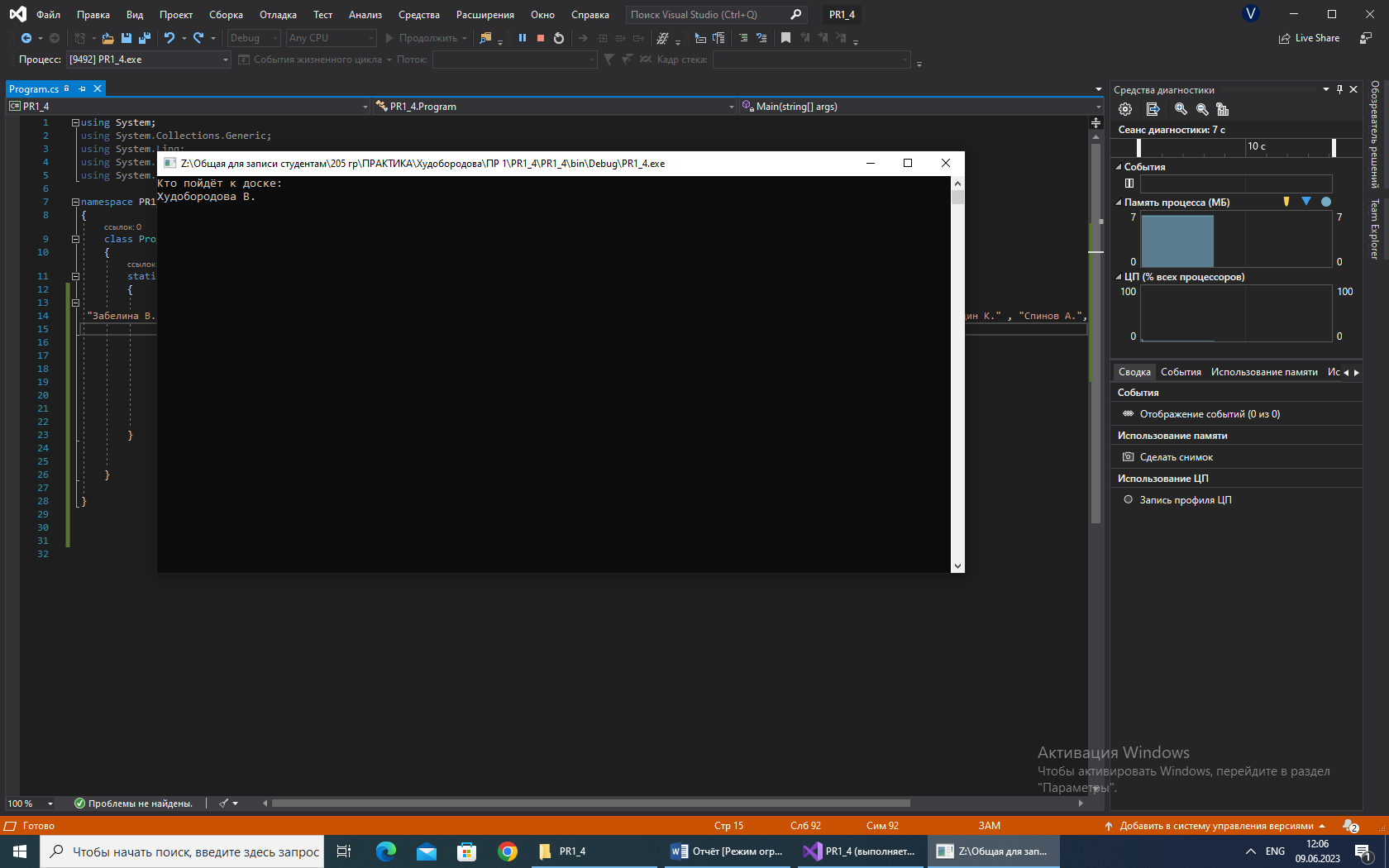


Рис. 4-Результат выполнения программы

Код программы:

class Program

{

static void Main(string[] args)

{

string[] people = { "Алиев Ф.", "Астанов А.", "Безрукова М.", "Борисова Е.", "Венков А.", "Говорова С.", "Гусев Д.", "Елисов Б.",

"Забелина В.", "Катин И.", "Кувакин Д" , "Кузьмина Юлиана.", "Лукьянчиков В.", "Маслов В.", "Осипов А.", "Петряшов И.", "Пушков Д.", "Сеткин В.", "Синицин К." , "Спинов А.", "Таранов С." ,

"Тюрина Е." , "Хорева А." , "Худобородова В." , "Шаров Э." , "Яненко Р." };

Random rand = new Random();

int numsName = rand.Next(people.Length);

Console.WriteLine("Кто пойдёт к доске:");

Console.WriteLine(people[numsName]);

Console.ReadKey();

}

Вывод: В ходе выполнения работы были освоены навыки написания массивов и программ, которые выдавали данные в рандомном порядке, в Visual Studio.

Практическая работа №2

Введение в XAML и WPF

Цель работы: В среде Microsoft Visual C# создать проект «Приложение WPF», в нём добавить кнопку и поменять фон.

Задание 1:

В среде Microsoft Visual C# создайте проект «Приложение WPF». Разместите в коде XAML в содержимом элемента Grid приведённый в методичке код.

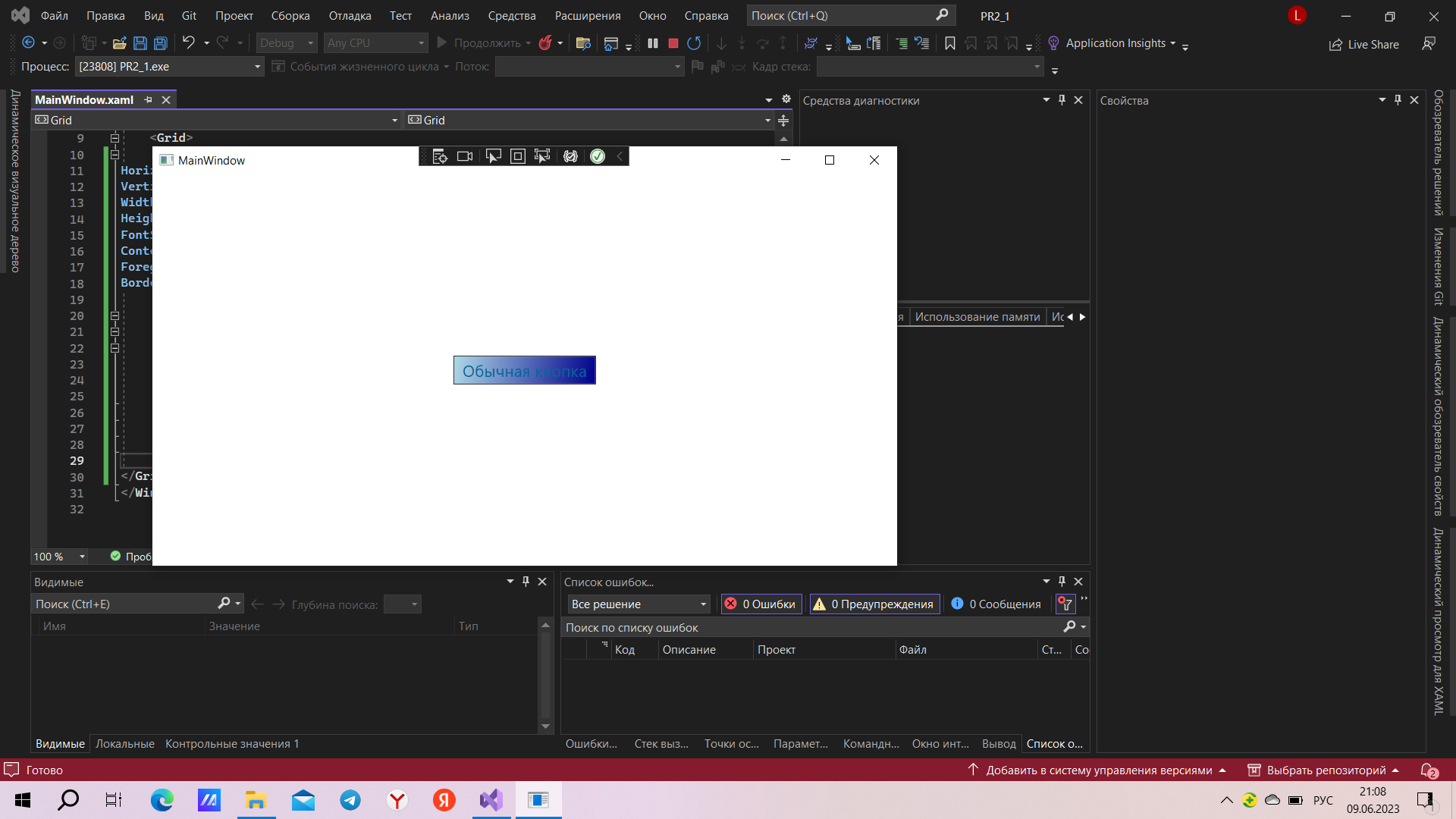


Рис.5 - Внешний вид данного приложения

Код программы:

<Window x:Class="PR2\_1.MainWindow"

<Grid>

<Button x:Name="Btn1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Width="150"

Height="30"

FontSize="17"

Content="Обычная кнопка"

Foreground="#006699"

BorderBrush="#303030" >

<Button.Background>

<LinearGradientBrush>

<LinearGradientBrush.GradientStops>

<GradientStop Color="LightBlue" Offset="0" />

<GradientStop Color="DarkBlue" Offset="1" />

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Button.Background>

</Button>

</Grid>

</Window>

Задание 2:

В XAML-коде для элемента Windows определите линейную градиентную заливку фона. Необходимо указать четыре промежуточные точки со смещениями 0, 0.2, 0.8 и 1.

Для задания вертикальной заливки необходимо определить атрибуты StartPoint и EndPoint для элемента LinearGradientBrush. Значения этих атрибутов указываются в формате “X,Y”, где X – относительное значение (от 0 до 1) абсциссы точки, Y – относительное значение (от 0 до 1) ординаты точки. Начало координат находится в левом верхнем углу окна. По умолчанию значения атрибутов StartPoint и EndPoint следующие: StartPoint=”0,0” EndPoint=”1,1”.

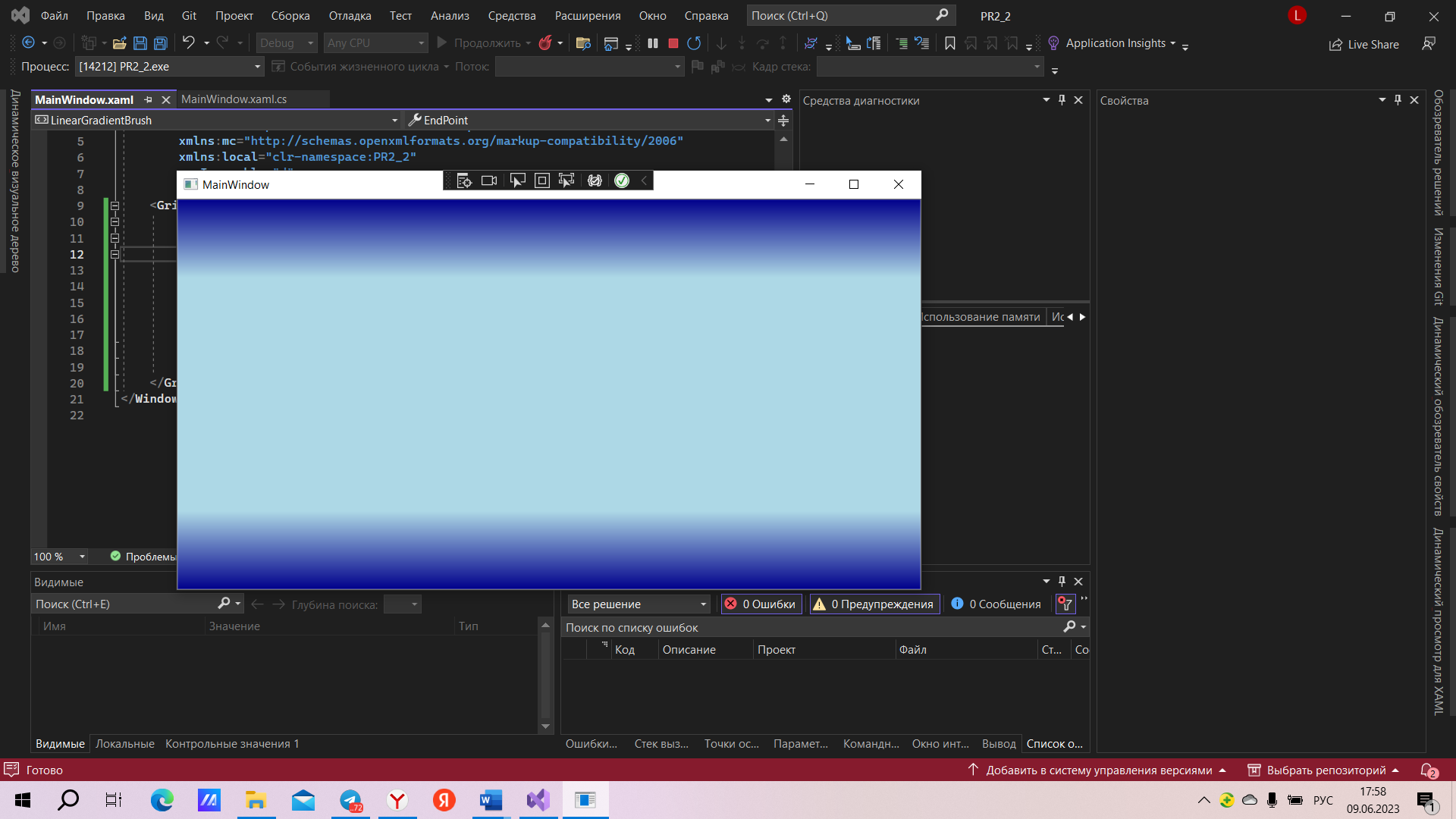


Рис. 6- Внешний вид данного приложения

Код программы:

<Window x:Class="PR2\_2.MainWindow"

<Grid>

<Button>

<Button.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="0,1">

<GradientStop Color="DarkBlue" Offset="0"/>

<GradientStop Color="LightBlue" Offset="0.2"/>

<GradientStop Color="LightBlue" Offset="0.8"/>

<GradientStop Color="DarkBlue" Offset="1"/>

</LinearGradientBrush>

</Button.Background>

</Button>

</Grid>

</Window>

Вывод: В ходе выполнения работы были освоены навыки создания проекта «Приложение WPF», а также научились добавлять кнопки, менять их цвет и, конечно же, менять цвет фона всего окна в целом.

# Практическая работа №3.

Диспетчеры компоновки

Цель работы: Ознакомиться с теоретической частью и разработать приложение WPF с графическим интерфейсом, который представлен в методичке.

Задание: Разработать приложение WPF c графическим интерфейсом

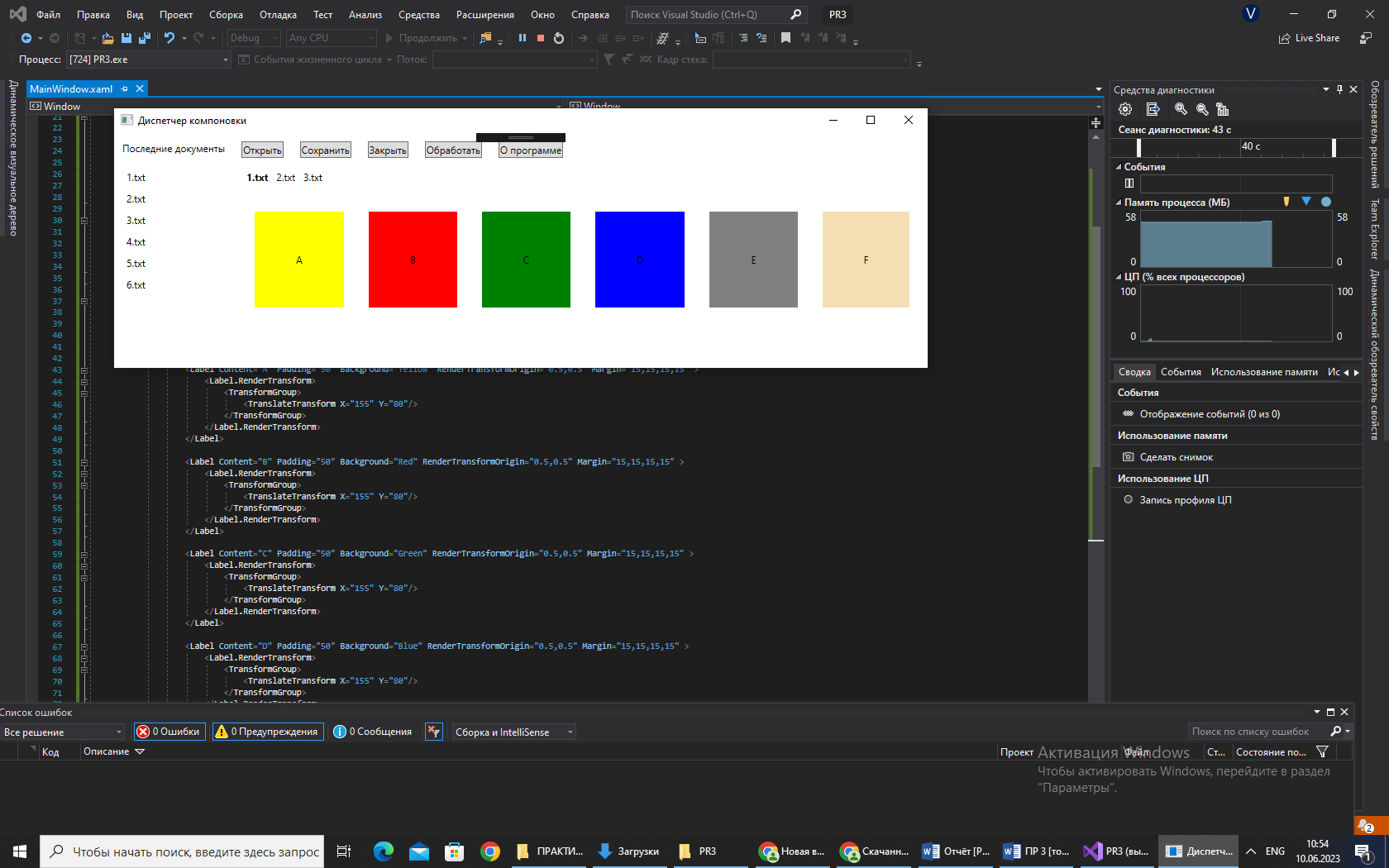


Рис. 7- Внешний вид данного приложения

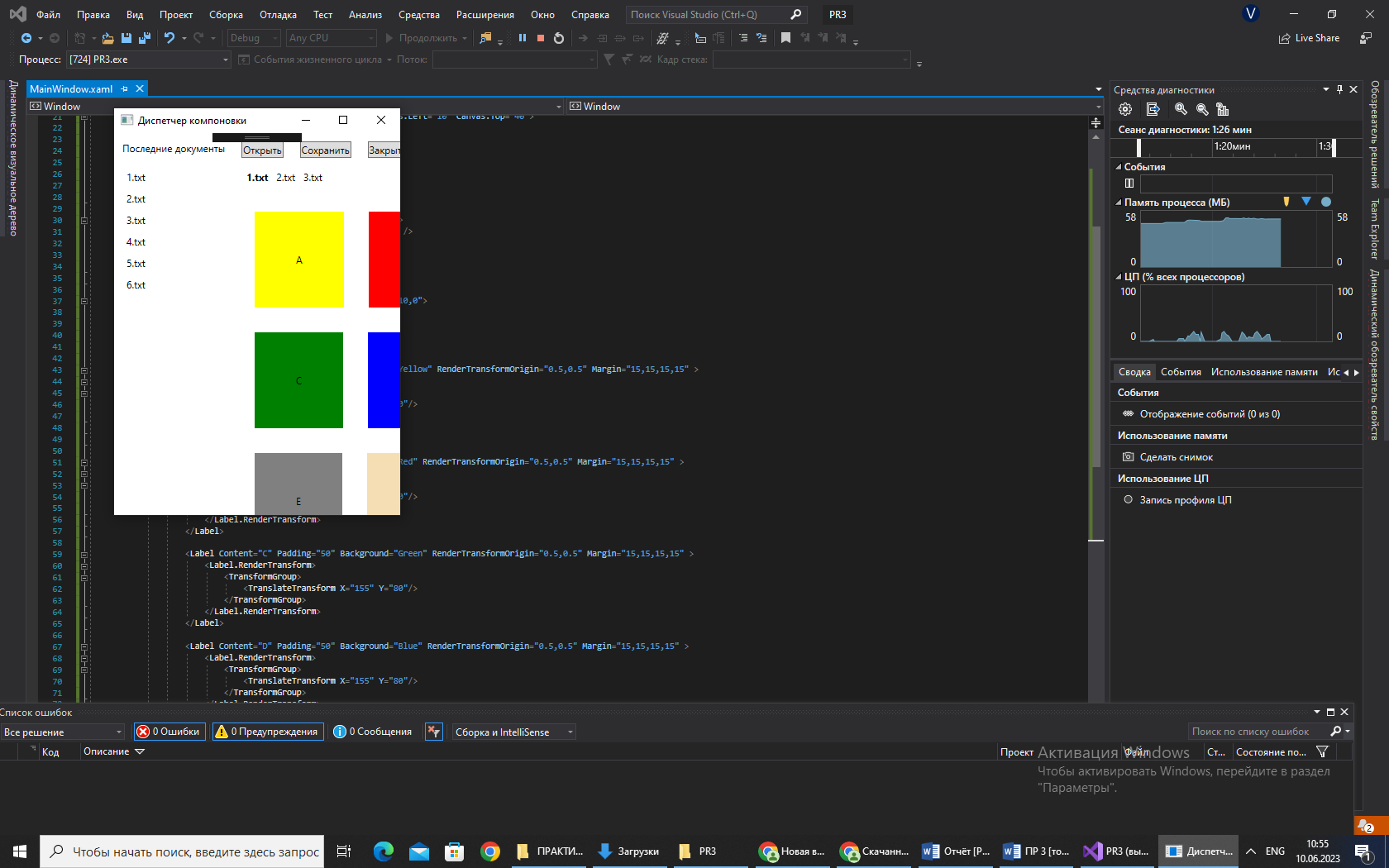


Рис. 8- Внешний вид, по мере уменьшения окна

Код программы:

<Window x:Class="PR3.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR3"

mc:Ignorable="d"

Title="Диспетчер компоновки" Height="407.282" Width="851.259">

<Grid>

<Canvas>

<WrapPanel>

<Label Content="Последние документы" Padding="10"/>

<Button Content="Открыть" Margin="10"/>

<Button Content="Сохранить" Margin="10"/>

<Button Content="Закрыть" Margin="10"/>

<Button Content="Обработать" Margin="10"/>

<Button Content="О программе" Margin="10"/>

</WrapPanel>

<StackPanel HorizontalAlignment="Left" Canvas.Left="10" Canvas.Top="40">

<Label Content="1.txt"/>

<Label Content="2.txt"/>

<Label Content="3.txt"/>

<Label Content="4.txt"/>

<Label Content="5.txt"/>

<Label Content="6.txt"/>

</StackPanel>

<WrapPanel Canvas.Left="155" Canvas.Top="40">

<Label Content="1.txt" FontWeight="Bold" />

<Label Content="2.txt"/>

<Label Content="3.txt"/>

</WrapPanel>

</Canvas>

<WrapPanel Orientation="Horizontal" Margin="0,0,10,0">

<Label Content="A" Padding="50" Background="Yellow" RenderTransformOrigin="0.5,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label Content="B" Padding="50" Background="Red" RenderTransformOrigin="0.5,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label Content="C" Padding="50" Background="Green" RenderTransformOrigin="0.5,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label Content="D" Padding="50" Background="Blue" RenderTransformOrigin="0.5,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label Content="E" Padding="50" Background="Gray" RenderTransformOrigin="1,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label Content="F" Padding="50" Background="Wheat" RenderTransformOrigin="0.5,0.5" Margin="15,15,15,15" >

<Label.RenderTransform>

<TransformGroup>

<TranslateTransform X="155" Y="80"/>

</TransformGroup>

</Label.RenderTransform>

</Label>

</WrapPanel>

</Grid>

</Window>

Вывод: В ходе выполнения работы, я ознакомилась с теоретической частью и разработала приложение WPF с графическим интерфейсом, который был представлен в методичке.

# Практическая работа №4

Основные элементы управления WPF

Цель работы: Ознакомиться с теоретической частью, ознакомиться с представленными там командами и выполнить задания.

Задание 1:

Разработать WPF-приложение с меню, панелью инструментов и строкой состояния. С помощью пунктов меню пользователь может изменять цвет фона окна, получить информацию о разработчике, а также закрыть окно. Кнопки панели инструментов дублируют команды меню. При наведении на пункты меню или кнопки панели инструментов в строке состояния отображается информация об этих элементах управления.

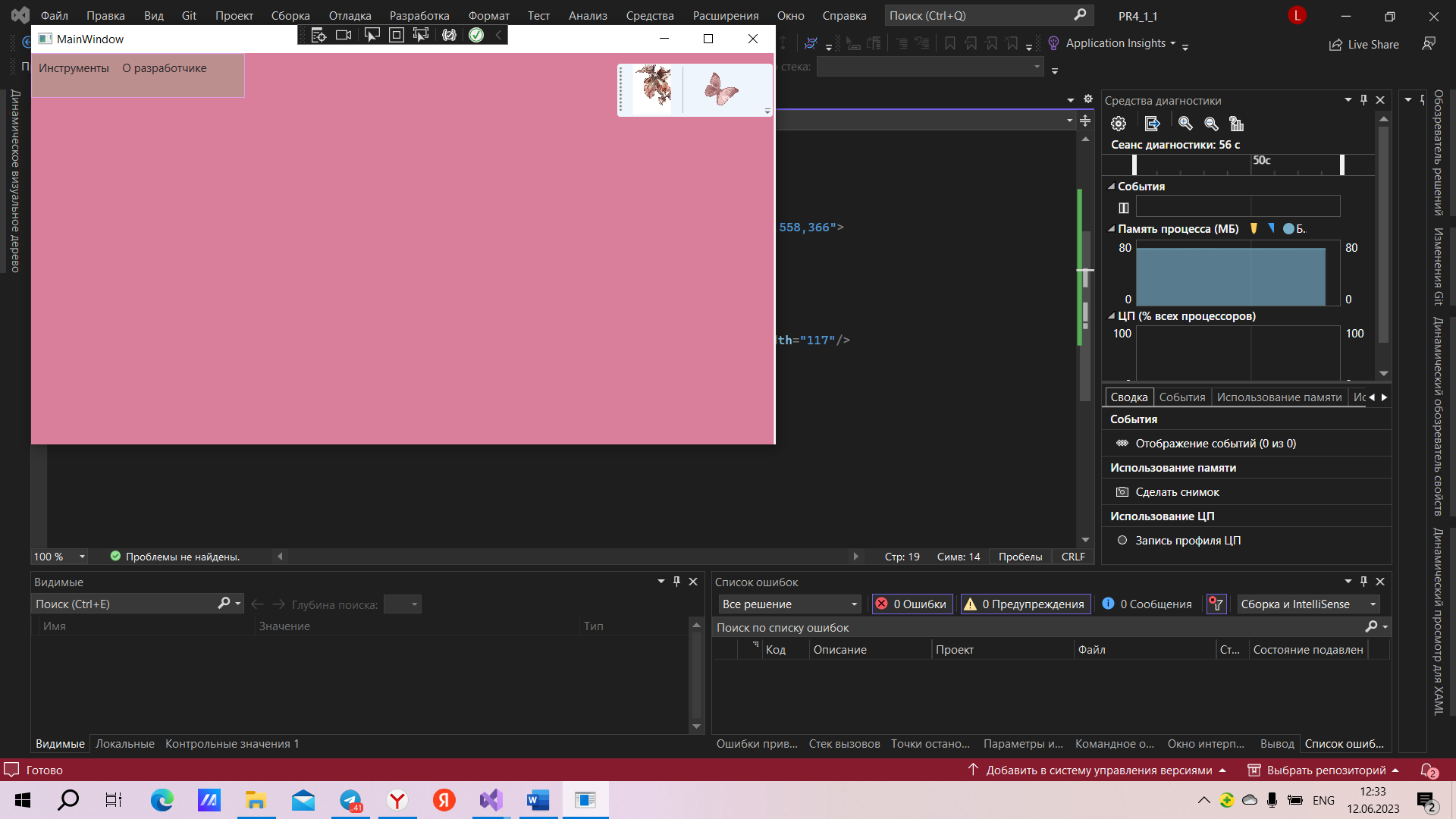


Рис.9- Внешний вид приложения

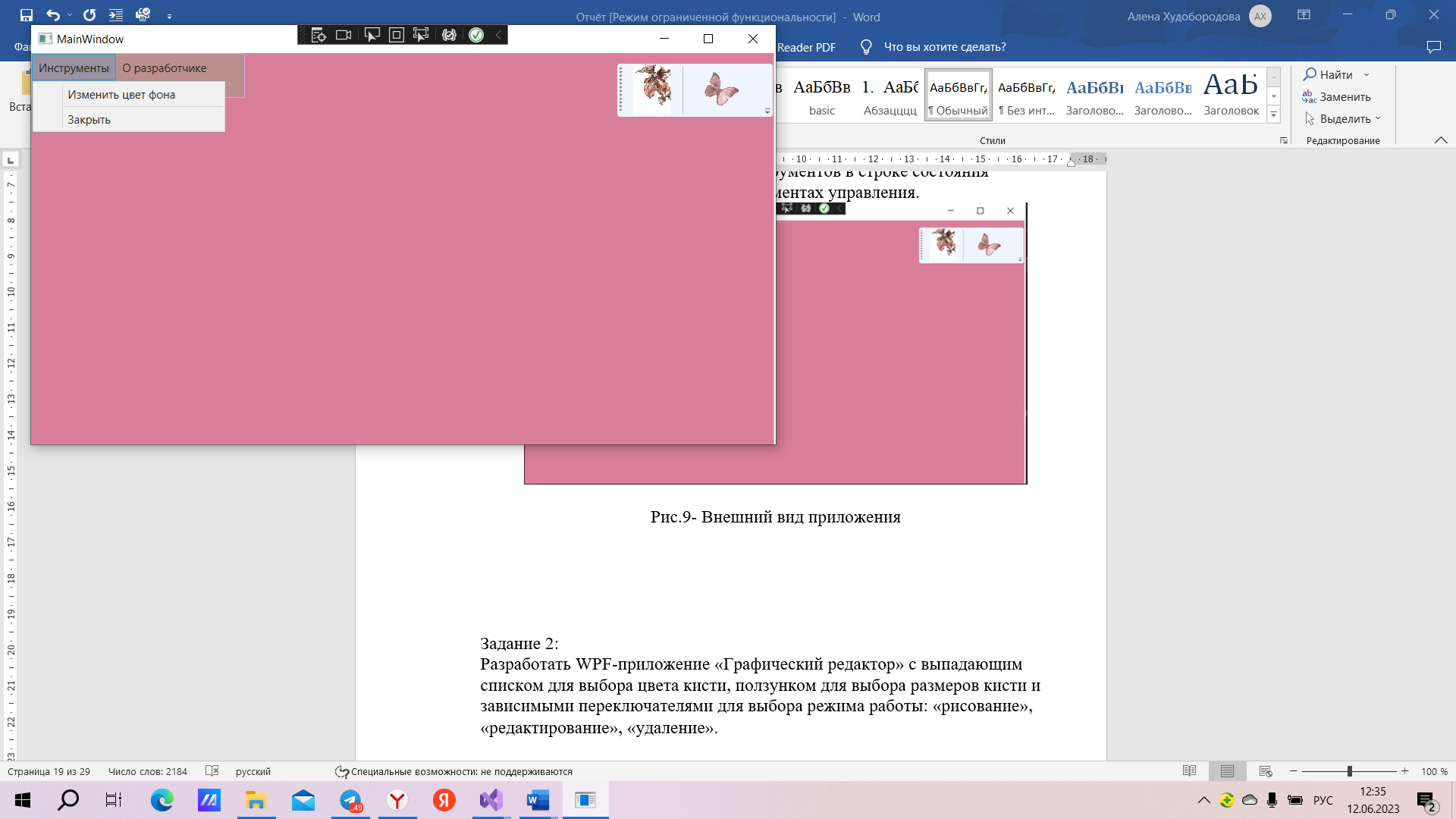


Рис. 10-Кнопки на панели приложения



Рис. 11- Информация из кнопки «О разработчике»

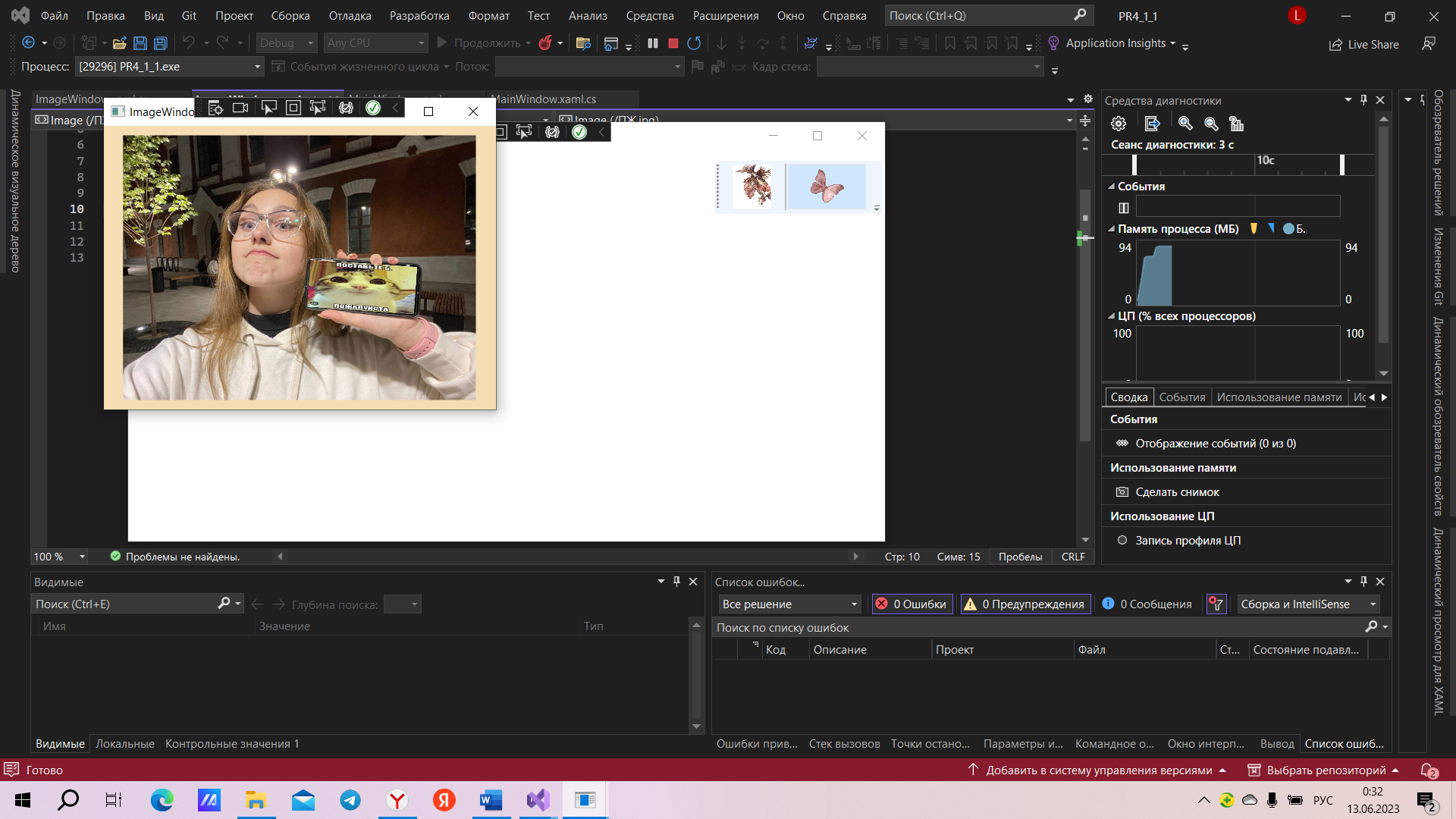


Рис. 12- Внешний вид приложения

Задание 2:

Разработать WPF-приложение «Графический редактор» с выпадающим списком для выбора цвета кисти, ползунком для выбора размеров кисти и зависимыми переключателями для выбора режима работы: «рисование», «редактирование», «удаление».

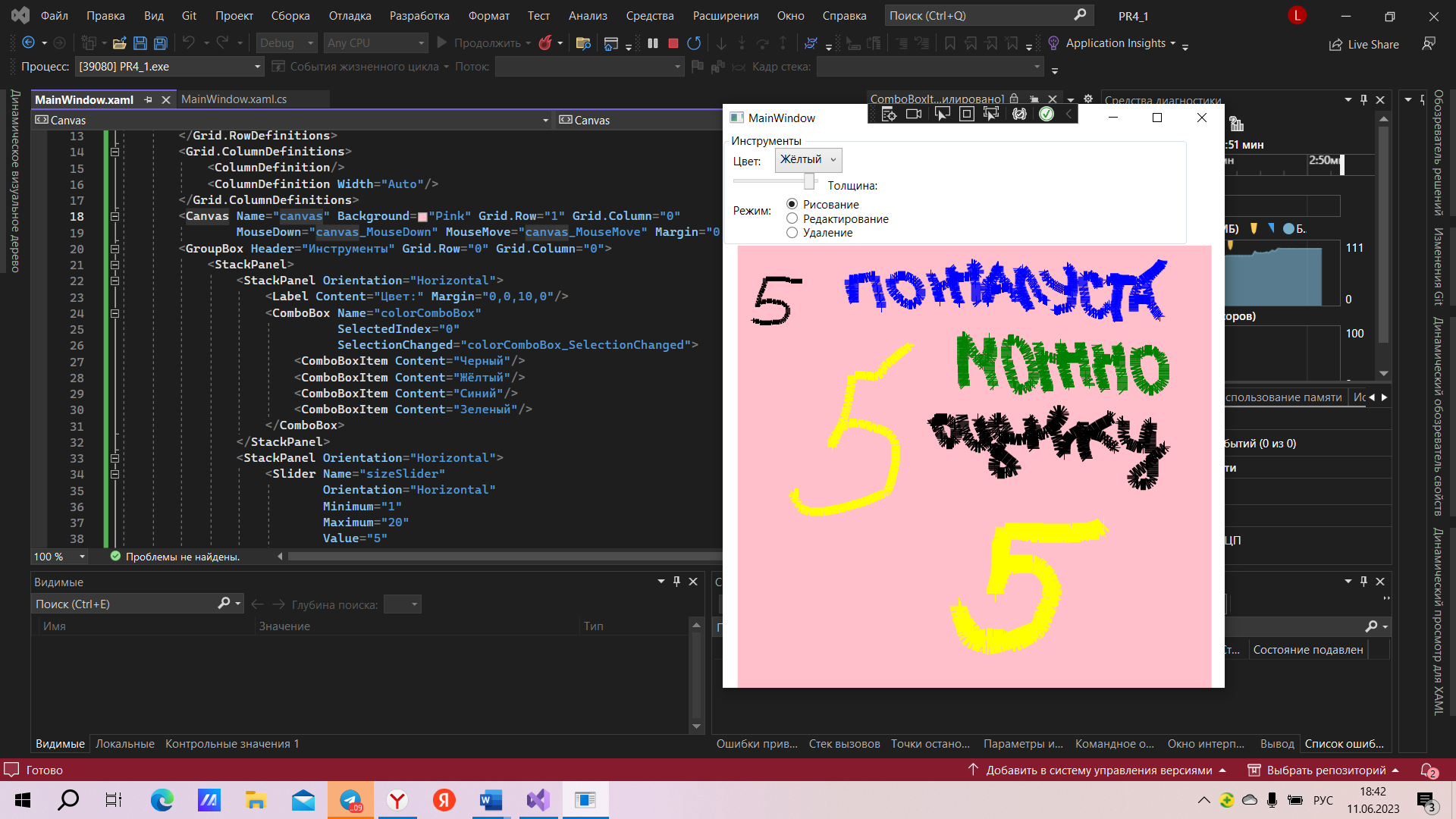


Рис.13- Внешний вид «Графического редактора»

Код программы MainWindow.xaml.cs:

namespace PR4\_1

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private SolidColorBrush brush = Brushes.Black;

private double canvasWidth = 500;

private double canvasHeight = 500;

private double brushSize = 5;

private Point startPoint;

private enum Mode { Drawing, Editing, Deleting };

private Mode mode = Mode.Drawing;

public MainWindow()

{

InitializeComponent();

canvas.Width = canvasWidth;

canvas.Height = canvasHeight;

}

private void colorComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

switch (colorComboBox.SelectedIndex)

{

case 0:

brush = Brushes.Black;

break;

case 1:

brush = Brushes.Yellow;

break;

case 2:

brush = Brushes.Blue;

break;

case 3:

brush = Brushes.Green;

break;

}

}

private void sizeSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

brushSize = sizeSlider.Value;

}

private void canvas\_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

startPoint = e.GetPosition(canvas);

if (mode == Mode.Deleting)

{

foreach (var shape in canvas.Children)

{

if (shape is Shape && ((Shape)shape).IsMouseOver)

{

canvas.Children.Remove((UIElement)shape);

break;

}

}

}

}

private void canvas\_MouseMove(object sender, System.Windows.Input.MouseEventArgs e)

{

if (e.LeftButton == System.Windows.Input.MouseButtonState.Pressed)

{

switch (mode)

{

case Mode.Drawing:

Line line = new Line();

line.Stroke = brush;

line.StrokeThickness = brushSize;

line.X1 = startPoint.X;

line.Y1 = startPoint.Y;

line.X2 = e.GetPosition(canvas).X;

line.Y2 = e.GetPosition(canvas).Y;

startPoint = e.GetPosition(canvas);

canvas.Children.Add(line);

break;

case Mode.Editing:

foreach (var shape in canvas.Children)

{

if (shape is Shape && ((Shape)shape).IsMouseOver)

{

Canvas.SetLeft((UIElement)shape, Canvas.GetLeft((UIElement)shape) + e.GetPosition(canvas).X - startPoint.X);

Canvas.SetTop((UIElement)shape, Canvas.GetTop((UIElement)shape) + e.GetPosition(canvas).Y - startPoint.Y);

startPoint = e.GetPosition(canvas);

break;

}

}

break;

case Mode.Deleting:

break;

}

}

}

private void drawingRadioButton\_Checked(object sender, RoutedEventArgs e)

{

mode = Mode.Drawing;

}

private void editingRadioButton\_Checked(object sender, RoutedEventArgs e)

{

mode = Mode.Editing;

}

private void deletingRadioButton\_Checked(object sender, RoutedEventArgs e)

{

mode = Mode.Deleting;

}

}

}

Код программы MainWindow.xaml:

<Window x:Class="PR4\_1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR4\_1"

mc:Ignorable="d"

Title="MainWindow" Height="508" Width="882">

<Grid Margin="0,0,39,53">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Canvas Name="canvas" Background="Pink" Grid.Row="1" Grid.Column="0"

MouseDown="canvas\_MouseDown" MouseMove="canvas\_MouseMove" Margin="0,0,-41,-50" Grid.ColumnSpan="2"/>

<GroupBox Header="Инструменты" Grid.Row="0" Grid.Column="0">

<StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Цвет:" Margin="0,0,10,0"/>

<ComboBox Name="colorComboBox"

SelectedIndex="0"

SelectionChanged="colorComboBox\_SelectionChanged">

<ComboBoxItem Content="Черный"/>

<ComboBoxItem Content="Жёлтый"/>

<ComboBoxItem Content="Синий"/>

<ComboBoxItem Content="Зеленый"/>

</ComboBox>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Slider Name="sizeSlider"

Orientation="Horizontal"

Minimum="1"

Maximum="20"

Value="5"

Width="100"

ValueChanged="sizeSlider\_ValueChanged"/>

<Label Content="Толщина:"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Режим:" Margin="0,0,10,0"/>

<StackPanel>

<RadioButton Content="Рисование"

IsChecked="True"

GroupName="mode"

Checked="drawingRadioButton\_Checked"/>

<RadioButton Content="Редактирование"

GroupName="mode"

Checked="editingRadioButton\_Checked"/>

<RadioButton Content="Удаление"

GroupName="mode"

Checked="deletingRadioButton\_Checked"/>

</StackPanel>

</StackPanel>

</StackPanel>

</GroupBox>

</Grid>

</Window>

Вывод: В ходе выполнения работы, я ознакомилась с теоретической частью и разработала приложение WPF с графическим интерфейсом, в котором можно порисовать разными цветами.

# Практическая работа №5

Привязка данных

Цель работы: Проверить реакцию среды разработки на неверные значения параметров, запустить приложение с следующим XAML-кодом и определить различие в поведении полей.

Задание 1:

Проверьте реакцию среды разработки на неверные значения параметров ElementName и Path. Проанализируйте сообщения, которые выводятся в окне вывода ( Вид-Вывод) при построение и при запуске приложения.

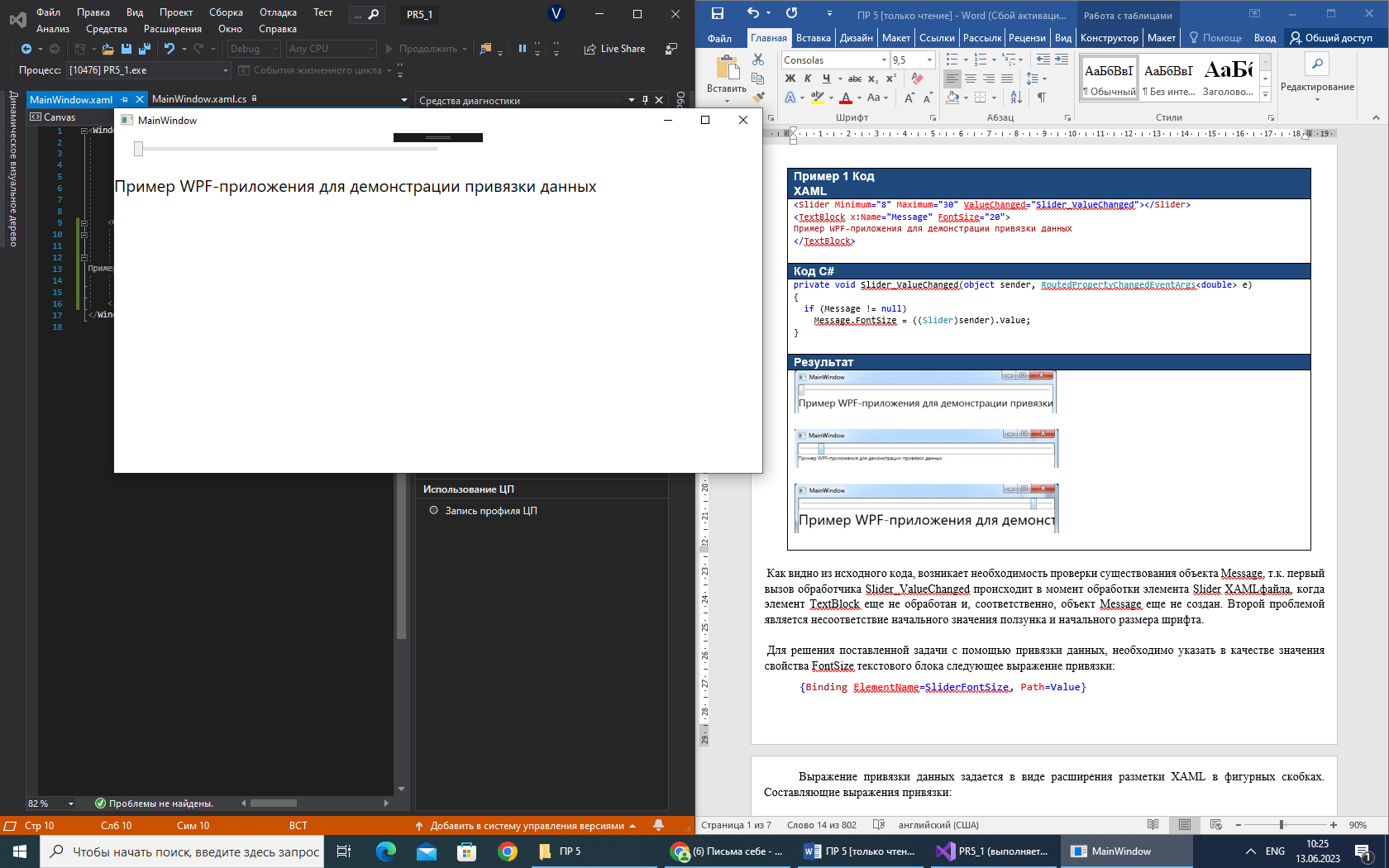


Рис. 14- Внешний вид приложения

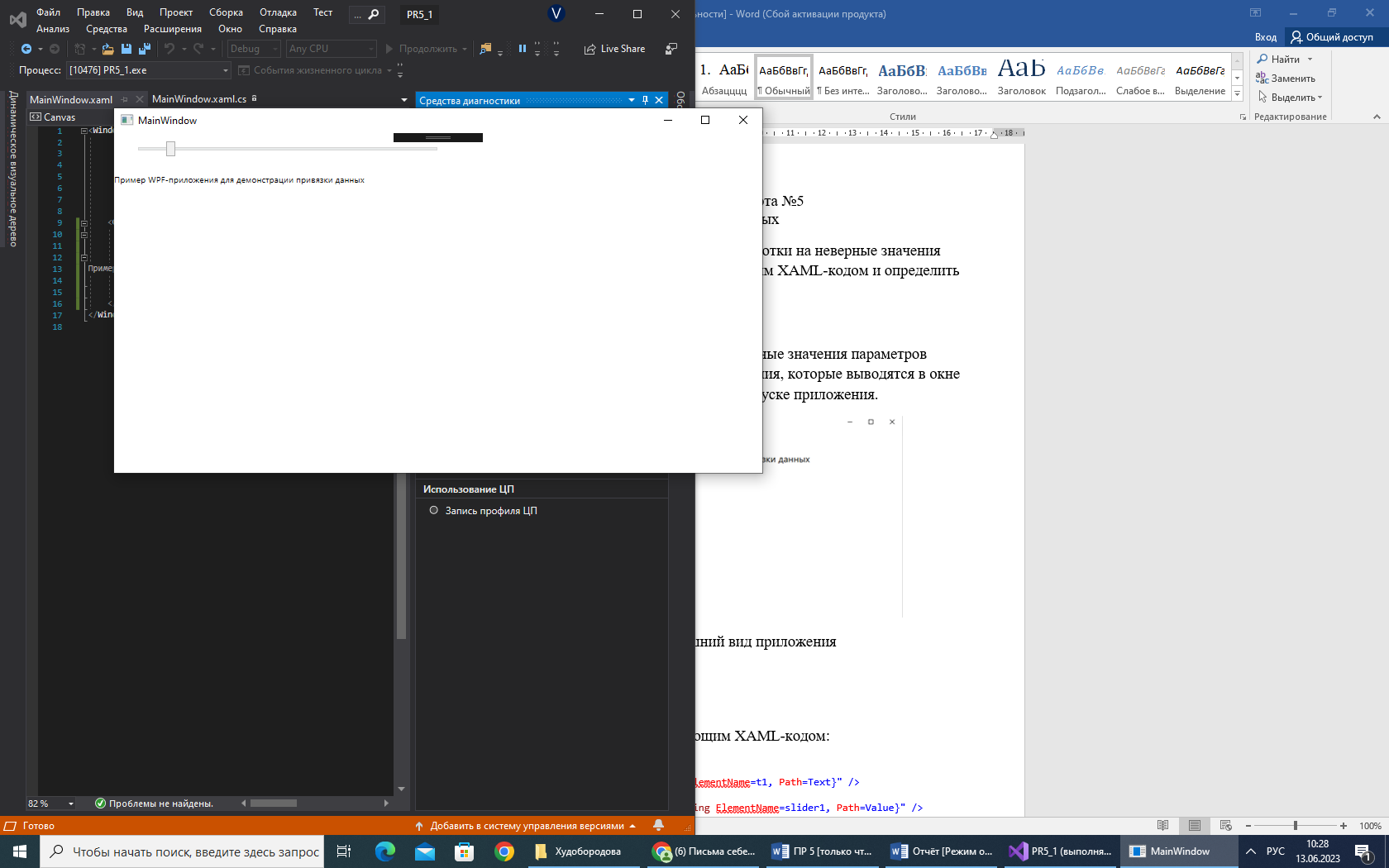


Рис. 15- Внешний вид приложения

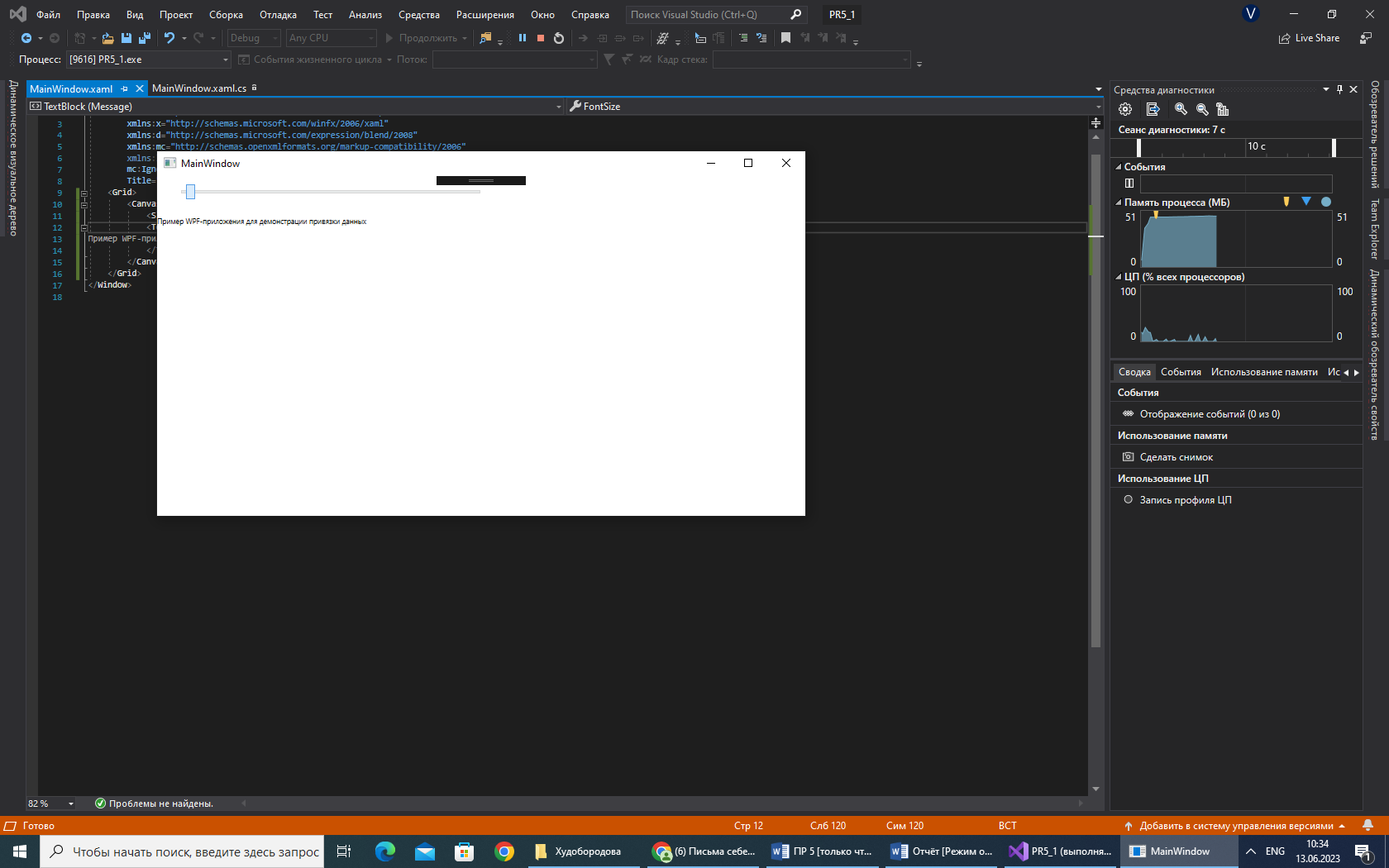


Рис. 16- Внешний вид приложения, после проверки реакции среды на ошибку.

Код программы MainWindow.xaml:

<Window x:Class="PR5\_1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR5\_1"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

<Canvas>

<Slider Minimum="8" Maximum="30" ValueChanged="Slider\_ValueChanged" Canvas.Left="24" Canvas.Top="10" Height="34" Width="372"></Slider>

<TextBlock x:Name="Message" Canvas.Top="49" Height="33" Width="590" FontSize="{Binding ElementName=SliderFontSize, Path=Value}">

Пример WPF-приложения для демонстрации привязки данных

</TextBlock>

</Canvas>

</Grid>

</Window>

Код программы MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace PR5\_1

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Slider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (Message != null)

Message.FontSize = ((Slider)sender).Value;

}

}

}

Задание 2:

Запустите приложение со следующим XAML-кодом:

<TextBox x:Name="t1" />

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text}" />

<Slider x:Name="slider1" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" />

Определите различие в поведении полей t1 и t2 и модифицируйте код, чтобы устранить это различие.

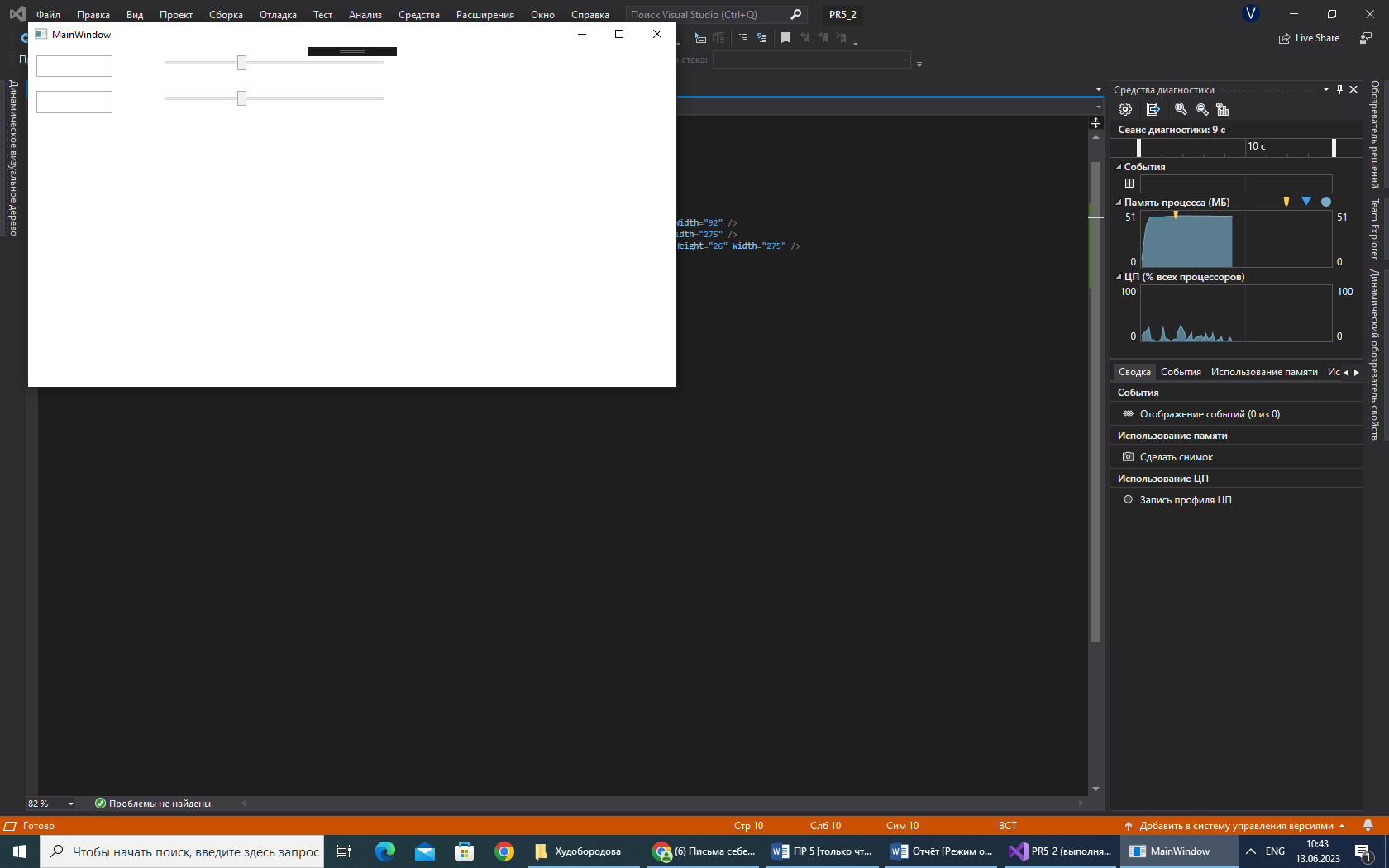


Рис.17- Внешний вид данного приложения

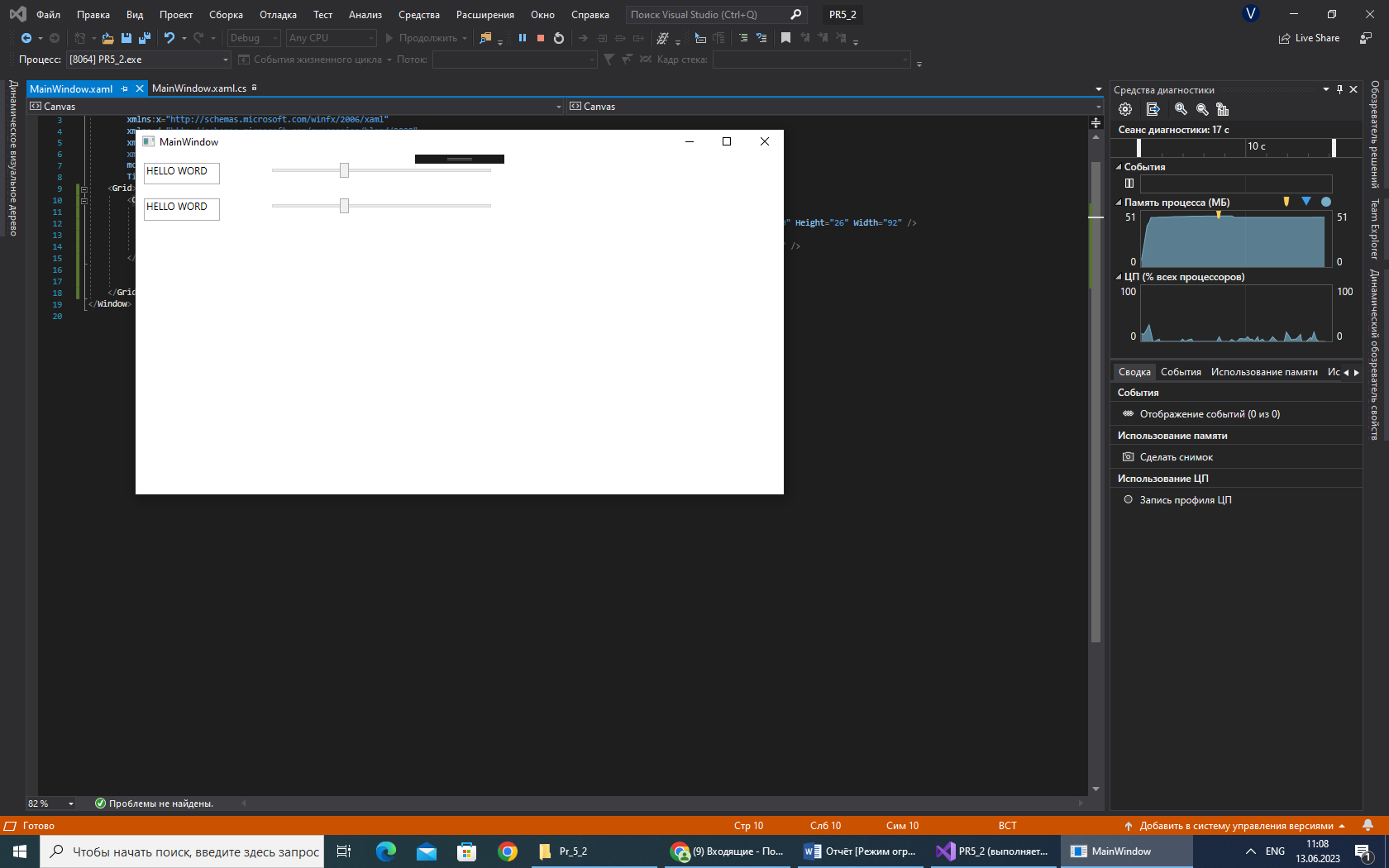


Рис. 18

Ответ: Различие полей t1 и t2 заключается в том, что при вводе текста в t1 значение t2 изменяется моментально, а при вводе текста в t2, значение t1 изменится только после того, как пользователь снимет выделение с t2

Чтобы исправить эту проблему необходимо t2 добавить свойство UpdateSourceTrigger и установить ему значение PropertyChanged, что означает, что значение будет изменяться моментально.

Код программы MainWindow.xaml:

<Window x:Class="PR5\_2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR5\_2"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

<Canvas>

<TextBox x:Name="t1" Canvas.Left="10" Canvas.Top="53" Height="27" Width="92" />

<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text, UpdateSourceTrigger=PropertyChanged}" Canvas.Left="10" Canvas.Top="10" Height="26" Width="92" />

<Slider x:Name="slider1" Canvas.Left="160" Canvas.Top="53" RenderTransformOrigin="-0.833,0.653" Height="27" Width="275" />

<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" Canvas.Left="160" Canvas.Top="10" Height="26" Width="275" />

</Canvas>

</Grid>

</Window>

Код программы MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace PR5\_2

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

Задание 3:

Дополните пример №2 текстовым полем ввода TextBox, в котором пользователь может ввести размер шрифта, и задайте выражения привязки таким образом, чтобы значение ползунка, текст текстового поля и размер шрифта текстового блока соответствовали друг другу.

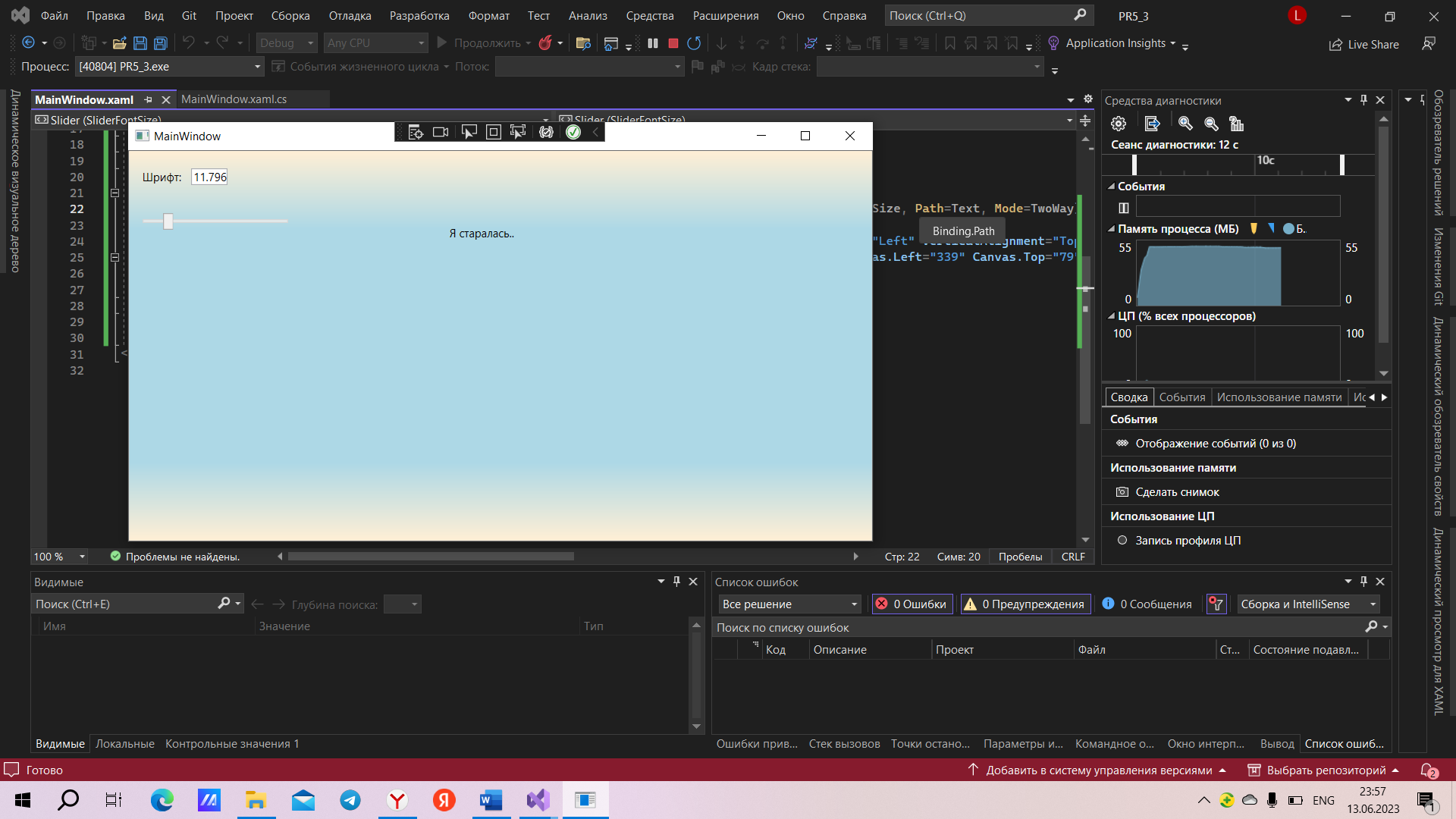


Рис. 19- Вид данного приложения

Код программы MainWindow.xaml:

<Window x:Class="PR5\_3.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR5\_3"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

<Button>

<Button.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="0,1">

<GradientStop Color="PapayaWhip" Offset="0"/>

<GradientStop Color="LightBlue" Offset="0.2"/>

<GradientStop Color="LightBlue" Offset="0.8"/>

<GradientStop Color="PapayaWhip" Offset="1"/>

</LinearGradientBrush>

</Button.Background>

</Button>

<Canvas>

<Slider x:Name="SliderFontSize" Minimum="8" Maximum="30" Value="{Binding ElementName=tb\_fontSize, Path=Text, Mode=TwoWay}" HorizontalAlignment="Left" VerticalAlignment="Top" Width="164" Canvas.Left="10" Canvas.Top="66"/>

<Label Canvas.Left="10" Canvas.Top="15">Шрифт:</Label>

<TextBox Text="5" x:Name="tb\_fontSize" Canvas.Left="66" Canvas.Top="19" HorizontalAlignment="Left" VerticalAlignment="Top" Width="39"></TextBox>

<TextBlock x:Name="Message" FontSize="{Binding ElementName=SliderFontSize, Path=Value}" Canvas.Left="339" Canvas.Top="79">

Я старалась..

</TextBlock>

</Canvas>

</Grid>

</Window>

Код программы MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace PR5\_3

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

Задание 4:

Модифицируйте приложения, разработанные в предыдущей лабораторной работе: удалите как можно больше обработчиков событий и реализуйте ту же функциональность приложения с помощью привязки данных.

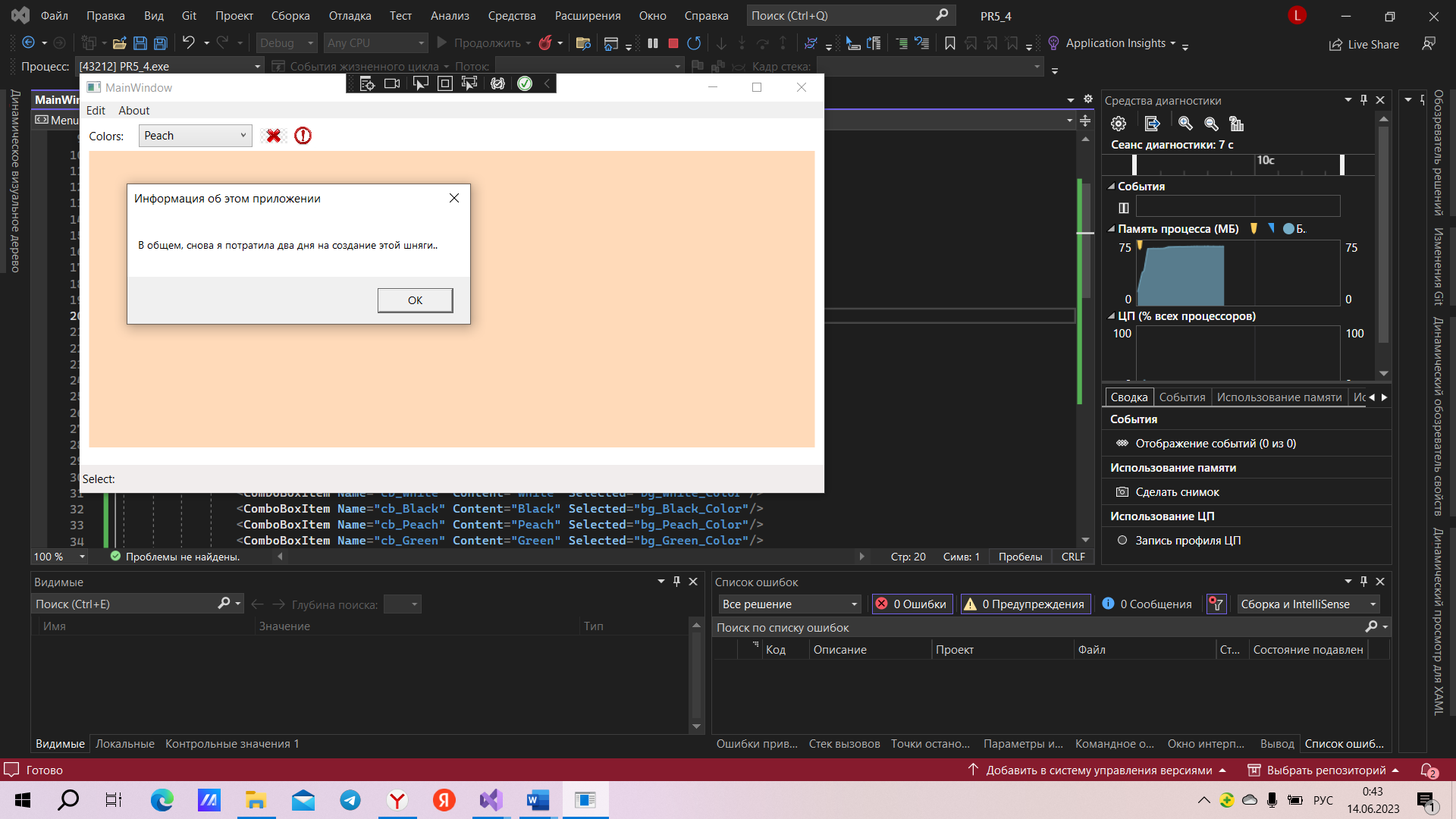


Рис.20- Модифицированное приложение из П.Р. 4

Код программы MainWindow.xaml:

<Window x:Class="PR5\_4.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PR5\_4"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

<Menu VerticalAlignment="Top">

<MenuItem x:Name="Edit" Header="Edit">

<MenuItem x:Name="mi\_colors" Header="Colors">

<RadioButton x:Name="mi\_White" Content="White" Checked="bg\_White\_Color"/>

<RadioButton x:Name="mi\_Black" Content="Black" Checked="bg\_Black\_Color"/>

<RadioButton x:Name="mi\_Peach" Content="Peach" Checked="bg\_Peach\_Color"/>

<RadioButton x:Name="mi\_Blue" Content="Blue" Checked="bg\_Blue\_Color"/>

<RadioButton x:Name="mi\_Yellow" Content="Yellow" Checked="bg\_Yellow\_Color"/>

</MenuItem>

<Separator/>

<MenuItem Header="Close window" Name="mi\_Close" Click="mi\_Close\_Click"/>

</MenuItem>

<MenuItem Header="About" Name="mi\_About" Click="mi\_About\_Click"/>

</Menu>

<WrapPanel VerticalAlignment="Top" Margin="0,18,0,0">

<TextBlock Text="Colors:" Margin="10"/>

<ComboBox x:Name="cb\_Colors" Margin="6" Width="120">

<ComboBoxItem Name="cb\_White" Content="White" Selected="bg\_White\_Color"/>

<ComboBoxItem Name="cb\_Black" Content="Black" Selected="bg\_Black\_Color"/>

<ComboBoxItem Name="cb\_Peach" Content="Peach" Selected="bg\_Peach\_Color"/>

<ComboBoxItem Name="cb\_Blue" Content="Blue" Selected="bg\_Blue\_Color"/>

<ComboBoxItem Name="cb\_Yellow" Content="Yellow" Selected="bg\_Yellow\_Color"/>

</ComboBox>

<Separator/>

<Button Click="mi\_Close\_Click" Background="Transparent" BorderBrush="Transparent">

<Image Source="/крест.png" Width="29" Height="18"/>

</Button>

<Button Click="mi\_About\_Click" Background="Transparent" BorderBrush="Transparent">

<Image Source="/инфа.jpg" Width="25"/>

</Button>

</WrapPanel>

<Rectangle x:Name="bg\_Rec" Margin="10,52,10,48"/>

<StatusBar Name="stBar" VerticalAlignment="Bottom" Height="30">

<TextBlock Text="Select: "/>

<TextBlock x:Name="st\_Hover\_el" Text=""/>

</StatusBar>

</Grid>

</Window>

Код программы MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace PR5\_4

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void bg\_Black\_Color(object sender, RoutedEventArgs e)

{

bg\_Rec.Fill = Brushes.Black;

mi\_Black.IsChecked = true;

cb\_Black.IsSelected = true;

}

private void bg\_White\_Color(object sender, RoutedEventArgs e)

{

bg\_Rec.Fill = Brushes.White;

mi\_White.IsChecked = true;

cb\_White.IsSelected = true;

}

private void bg\_Peach\_Color(object sender, RoutedEventArgs e)

{

bg\_Rec.Fill = Brushes.PeachPuff;

mi\_Peach.IsChecked = true;

cb\_Peach.IsSelected = true;

}

private void bg\_Blue\_Color(object sender, RoutedEventArgs e)

{

bg\_Rec.Fill = Brushes.Blue;

mi\_Blue.IsChecked = true;

cb\_Blue.IsSelected = true;

}

private void bg\_Yellow\_Color(object sender, RoutedEventArgs e)

{

bg\_Rec.Fill = Brushes.Yellow;

mi\_Yellow.IsChecked = true;

cb\_Yellow.IsSelected = true;

}

private void mi\_Close\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void mi\_About\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("В общем, снова я потратила два дня на создание этой шняги..", "Информация об этом приложении");

}

}

}

Вывод: В ходе выполнения работы, я проверила реакцию среды разработки на неверные значения параметров, выяснилось, что если изменить код, то запускаться само приложение будет не с теми параметрами. Запустила приложение с XAML-кодом, который был представлен в методичке и проверила поведение полей, что если вводить текст в одной строчке, он автоматом пишет её в другой, также размер меняется вместе. В третьем задании, я установила параметры, где можно задать самим шрифт и ползунок встанет сразу на это место, а сам текст, естественно поменяется в размере в указанных параметрах, а также в четвёртом задание модифицировали приложения из прошлой практической.

# Практическая работа №6

Использование стилей в WPF-приложениях

Цель работы: Ознакомиться с теорией в методичке, проверить, какое значение имеет больший приоритет: значение свойства, указанное в стиле, или значение атрибута элемента, модифицировать WPF-приложение, разработанное в 3-ей лабораторной работе; Разработать приложение MultiEdit для одновременной работы с несколькими текстами.

Задание 1:

Проверьте, какое значение имеет больший приоритет: значение свойства, указанное в стиле, или значение атрибута элемента.

Задание 2:

Модифицируйте WPF-приложение, разработанное в 3-ей лабораторной работе: используйте стили для однотипных элементов управления.

Задание 3:

Разработайте приложение MultiEdit для одновременной работы с несколькими текстами. Окно должно быть разделено на две части с одинаковыми градиентами. В каждой части окна должно быть несколько многострочных текстовых полей: одно из них большого размера с крупным шрифтом, а остальные маленького размера с мелким шрифтом. То текстовое окно, в котором пользователь набирает текст, должно быть большим, остальные текстовые поля должны быть маленькими. Внешний вид однотипных элементов управления должен определяться с помощью стилей.

# Практическая работа №7

Триггеры в WPF-приложениях

Цель работы: Ознакомиться с теорией, представленной в методичке и выполнить задания.

Задание 1:

Рассмотрите случай, когда для одного и того же элемента управления срабатывают сразу несколько триггеров, устанавливающих для одного и того же свойства различные значения, и определите правило, по которому определяется приоритет применения элементов Setter этих триггеров.

Задание 2:

Разработайте WPF-приложение с двумя многострочными текстовыми полями, кнопками «Открыть», «Очистить», «Закрыть» и выпадающим списком для задания внешнего вида текстовых полей. Задайте для текстовых полей одинаковый градиентный фон. Кнопка «Закрыть» должна быть доступна только в том случае, если в обоих текстовых полях нет текста. Задайте для кнопок различный внешний вид при наведении курсора и при нажатии на них. Внешний вид текстовых полей (тип шрифта, размер шрифта, цвет шрифта) должен меняться в зависимости от значения, выбранного в выпадающем списке.

Вывод:

# Практическая работа №8

Использование кистей в WPF-приложениях

Цель работы: Ознакомиться с теорией, представленной в методичке и выполнить задание.

Задание:

На основе примера 8 разработайте WPF-приложение с анимацией источника света, светового пятна от него на шаре и отражения шара. Для анимации начала радиального градиента используйте элемент PointAnimation, свойства From и To которого задаются в формате ”X,Y” (пример: To=”0,1”)

Вывод:

# Заключение