Содержание

[Введение 2](#_Toc137295370)

[Техника безопасности при работе за компьютером 3](#_Toc137295371)

[Практическая работа № 1 5](#_Toc137295372)

[Практическая работа № 2 11](#_Toc137295373)

[Практическая работа № 3 13](#_Toc137295374)

Введение

В ходе выполнения практических работ необходимо: Ознакомиться с процессом анализа предметной области и получить навыки по использованию методов анализа предметной области; освоить технологии документирования программных средств (ПС) на начальных стадиях проектирования в соответствии с ЕСПД; реализовать начальные этапов процесса разработки программного средства в соответствии с ГОСТ Р ИСО/МЭК 12207; ознакомиться с методами проектирования архитектуры, построением концептуальных моделей и типами моделей данных; ознакомиться с созданием информационных систем и выполнить конфигурацию созданного проекта; разработать АИС для работы с базами данных; протестировать созданное ранее приложение.

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 поводах, 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;

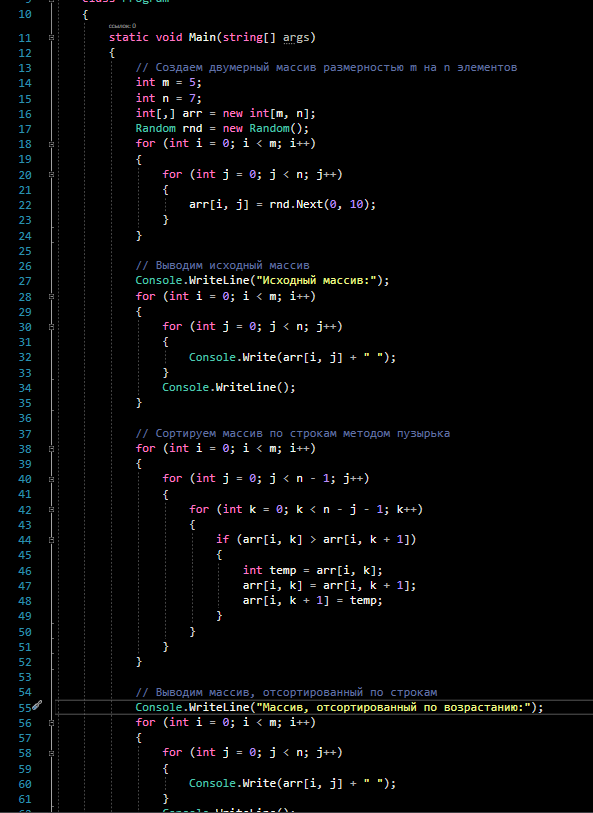
Практическая работа № 1

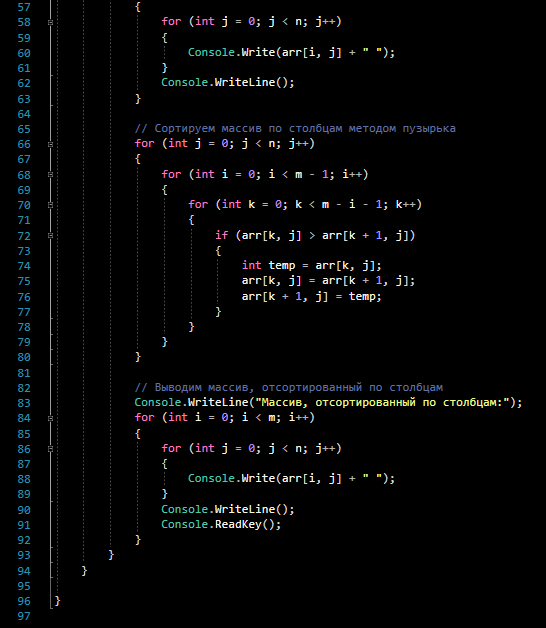
Анализ выбранной предметной области

Цель работы: Ознакомится с процессом создания массива и получить навыки работы с ним.

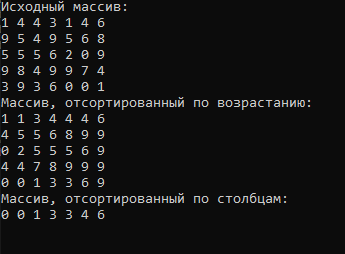
Задание 1: Нужно было создать двумерный массив размерностью m на n элементов, заполнив его случайными числами в диапазоне от 0 до 9. Отсортировать элементы массива по возрастанию вначале по строкам, а затем по столбцам. Вывести на экран исходный массив, массива отсортированный построчно, массив отсортированный по столбцам. Для сортировки применить метод пузырька

Код программы:





Работа кода:

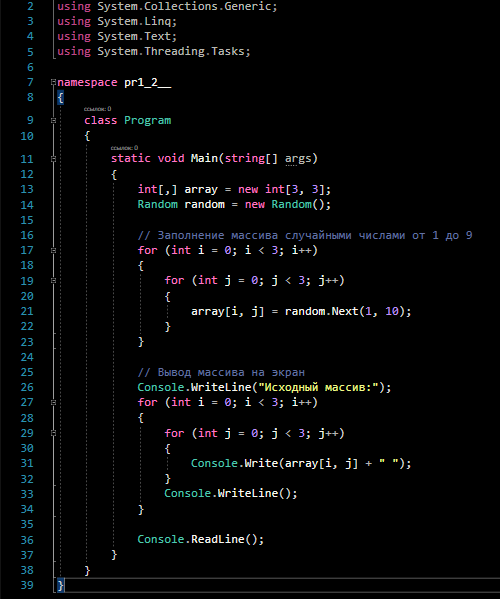


Как работает это:

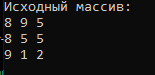
1.В этом примере мы используем вложенные циклы для создания двумерного массива и заполнения его случайными числами. Затем мы выводим исходный массив на экран.  
2.Далее мы сортируем массив по строкам методом пузырька, используя вложенные циклы. Мы проходим по каждой строке и сравниваем пары соседних элементов, меняя их местами, если они находятся в неправильном порядке.  
3.Затем мы выводим отсортированный массив по строкам на экран.  
4.Затем мы сортируем массив по столбцам методом пузырька, используя вложенные циклы. Мы проходим по каждому столбцу и сравниваем пары соседних элементов, меняя их местами, если они находятся в неправильном порядке.  
5.Наконец, мы выводим отсортированный массив по столбцам на экран.

Задание 2: Нужно было создать двумерный массив размерности 3x3, и заполнить его числами от 1 до 9, расположенными случайным образом

Код программы:



Работа кода:



Как работает код

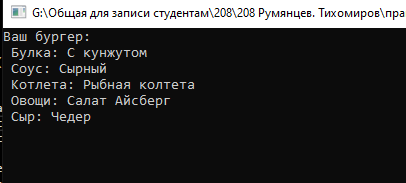
1.В этом примере мы используем класс: Random для генерации случайных чисел от 1 до 9.

2.Затем мы используем два вложенных цикла: for для заполнения массива и вывода его на экран.

Задание 3: Нужно было создать программу конструктора бургеров, каждый запуск которой формирует случайный рецепт из возможных ингредиентов

Код программы: 

Работа кода:



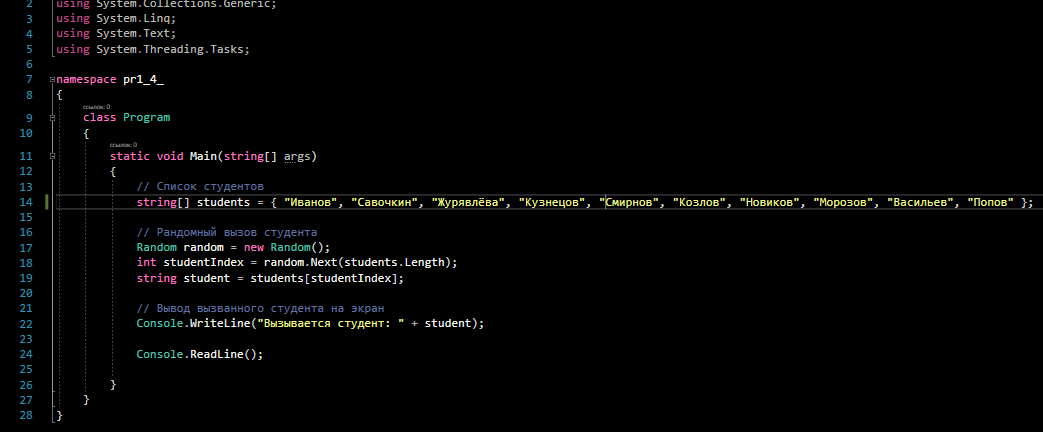
Как работает код

1.В этом примере мы используем массив: ingredients, содержащий возможные ингредиенты для бургеров.

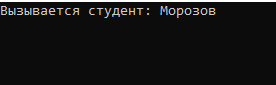
2.Затем мы используем класс: Random для формирования случайного рецепта, выбирая случайные ингредиенты из массива ingredients.

3.Длина рецепта выбирается случайным образом, после выполнения программы нам выводят рандомные ингредиенты.

Задание 4: Написать программу рандомного вызова студентов своей группы.

Код работы: 

Работа кода:



Как работает код:

В этом примере мы используем массив students, содержащий список студентов в группе. Затем мы используем класс Random для выбора случайного студента из массива students. Наконец, мы выводим вызванного студента на экран.

Вывод: В ходе выполнения практической работы было вспомнили как работать с массивами ,а так же закреплён материал работы с рандомом.

# Практическая работа № 2

Введение в XAML и WPF

Цель работы: Получить начальные знания об архитектуре WPF, подходах к проектированию интерфейса приложений, создать приложение WPF, в XAML-коде для элемента Windows определить линейную градиентную заливку фона.

Задание 1:

В среде Microsoft Visual C# создайте проект «Приложение WPF». Разместить в коде XAML в содержимом элемента Grid следующий код:

<Button x:Name="Btn1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Width="150"

Height="30"

FontSize="17"

Content="Обычная кнопка"

Foreground="#006699"

Background="#f0f0f0"

BorderBrush="#303030" />

Запустить приложение и проверьте его поведение при изменении размеров окна.

<Window x:Class="Задание\_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:Задание\_1"

mc:Ignorable="d"

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

<Grid>

<Button x:Name="Btn1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Width="150"

Height="30"

FontSize="17"

Content="Обычная кнопка"

Foreground="#006699"

Background="#f0f0f0"

BorderBrush="#303030" />

</Grid>

</Window>

Задание 2:

В XAML-коде для элемента Windows определите линейную градиентную заливку фона в соответствии с рисунком: 

Вывод: В ходе выполнения практической работы было произведено ознакомление с архитектурой WPF, подходах к проектированию интерфейса приложений, создано само приложение WPF и в XAML-коде определена линейная градиентная заливка фона.

# Практическая работа № 3

**«Диспетчеры компоновки»**

Цель работы: Изучить компоновку Canvas,WrepPanel,StackPanel,DockPanel и Grid.Разработать графический интерфейс.

Графический интерфейс:

Код который использовали: <Grid>

<Canvas>

<Label Canvas.Left="10" Canvas.Top="5" Content="Последние документы" />

<Label Canvas.Left="10" Canvas.Top="5" Content="Последние документы" />

<Button Canvas.Left="155" Canvas.Top="10" Content="Открыть" />

<Button Canvas.Left="214" Canvas.Top="10" Content="Сохранить" />

<Button Canvas.Left="285" Canvas.Top="10" Content="Закрыть" />

<Button Canvas.Left="343" Canvas.Top="10" Content="Обработать" />

<Button Canvas.Left="420" Canvas.Top="10" Content="О программе" />

<Label Canvas.Left= "10" Canvas.Top="54" Content="Отчёт.txt"/>

<Label Canvas.Left= "10" Canvas.Top="72" Content="Задание.txt"/>

<Label Canvas.Left= "10" Canvas.Top="90" Content="3.txt"/>

<Label Canvas.Left= "10" Canvas.Top="116" Content="4.txt"/>

<Label Canvas.Left= "10" Canvas.Top="134" Content="Ответы.txt"/>

<Label Canvas.Left= "10" Canvas.Top="165" Content="Текст.txt"/>

<Label Canvas.Left= "150" Canvas.Top="30" Content="1.txt"/>

<Label Canvas.Left= "150" Canvas.Top="30" Content="1.txt"/>

<Label Canvas.Left= "180" Canvas.Top="30" Content="2.txt"/>

<Label Canvas.Left= "210" Canvas.Top="30" Content="3.txt"/>

<Rectangle

Width="55"

Height="60"

Fill="Yellow"

Canvas.Left="165"

Canvas.Top="58" />

<Label Canvas.Left= "185" Canvas.Top="73" Content="A" />

<Rectangle

Width="55"

Height="60"

Fill="Red"

Canvas.Left="165"

Canvas.Top="134" RenderTransformOrigin="-0.399,1.619" />

<Label Canvas.Left= "185" Canvas.Top="147" Content="B" RenderTransformOrigin="0.529,0.434" />

<Rectangle

Width="55"

Height="60"

Fill="Green"

Canvas.Left="165"

Canvas.Top="214" />

<Label Canvas.Left= "185" Canvas.Top="233" Content="C" RenderTransformOrigin="0.592,0.548" />

<Rectangle

Width="55"

Height="60"

Fill="Blue"

Canvas.Left="375"

Canvas.Top="58" />

<Label Canvas.Left= "395" Canvas.Top="73" Content="D" />

<Rectangle

Width="55"

Height="60"

Fill="Gray"

Canvas.Left="445"

Canvas.Top="58" />

<Label Canvas.Left= "465" Canvas.Top="73" Content="E" />

<Rectangle

Width="55"

Height="60"

Fill="LightGoldenrodYellow"

Canvas.Left="515"

Canvas.Top="58" />

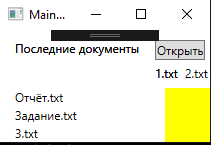
<Label Canvas.Left= "535" Canvas.Top="73" Content="F" />

</Canvas>

</Grid>

</Window>

Работа кода:





Вывод по работе: я научился создавать программу на платформе WPF, написал графическую программу, научился фиксировать значки и при изменении размера окна работает как надо.

Практическая работа № 4

«Основные элементы управления WPF»

Цель работы: Ознакомиться с основными элементами управления WPF и применить их на практике.

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.

Код готовой программы 1:

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using CD = System.Windows.Forms.ColorDialog;

using DR = System.Windows.Forms.DialogResult;

namespace Задание\_1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void ChangeColor\_Click(object sender, RoutedEventArgs e)

{

CD color = new CD();

DR result = color.ShowDialog();

if (result == DR.OK)

Background = new SolidColorBrush(Color.FromRgb(color.Color.R, color.Color.G, color.Color.B));

}

private void Developers\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Данное приложение было разработано: Гарелин Егор");

}

private void Close\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void MEnter(object sender, MouseEventArgs e)

{

StatusLabel.Content = ((FrameworkElement)sender).Tag;

}

private void MLeave(object sender, MouseEventArgs e)

{

StatusLabel.Content = null;

}

}

}

Код готовой программы 2:

<Window x:Class="Задание\_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:Задание\_1"

mc:Ignorable="d"

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

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Menu Height="50" VerticalAlignment="Top">

<MenuItem MouseLeave="MLeave" MouseEnter="MEnter" Tag="Сменить цвет фона окна" Header="Цвет окна" Click="ChangeColor\_Click" />

<MenuItem MouseLeave="MLeave" MouseEnter="MEnter" Tag="Информация о разработчиках приложения" Header="Разработчики" Click="Developers\_Click" />

<MenuItem MouseLeave="MLeave" MouseEnter="MEnter" Tag="Закрыть окно" Header="Закрыть окно" Click="Close\_Click" />

</Menu>

<ToolBar Grid.Column="1" Height="50" VerticalAlignment="Top">

<Button MouseLeave="MLeave" MouseEnter="MEnter" Tag="Сменить цвет фона окна" Click="ChangeColor\_Click">

<Image Source="https://cdn.icon-icons.com/icons2/624/PNG/512/Paint\_Palette-80\_icon-icons.com\_57294.png"/>

</Button>

<Button MouseLeave="MLeave" MouseEnter="MEnter" Tag="Информация о разработчиках приложения" Click="Developers\_Click">

<Image Source="https://cdn.icon-icons.com/icons2/1865/PNG/512/programmer\_119550.png"/>

</Button>

<Button MouseLeave="MLeave" MouseEnter="MEnter" Tag="Закрыть окно" Click="Close\_Click">

<Image Source="https://cdn.icon-icons.com/icons2/2853/PNG/512/delete\_remove\_close\_icon\_181533.png"/>

</Button>

</ToolBar>

<StatusBar VerticalAlignment="Bottom" Height="25" Grid.ColumnSpan="2">

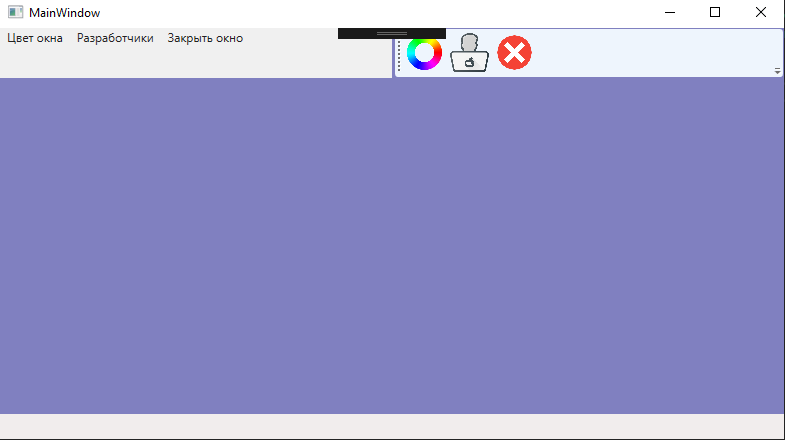
<Label Padding="0" Name="StatusLabel"/>

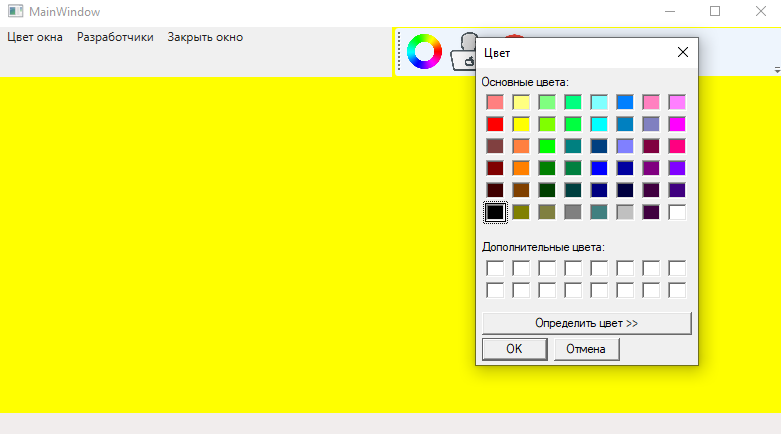
</StatusBar>

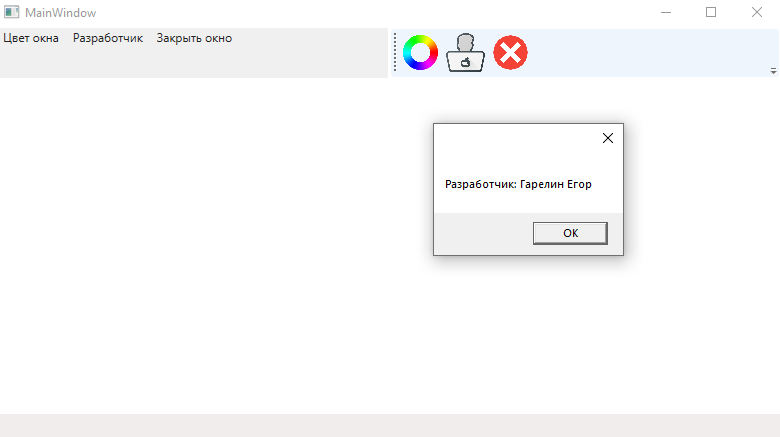
</Grid>

</Window>

Работа кода:

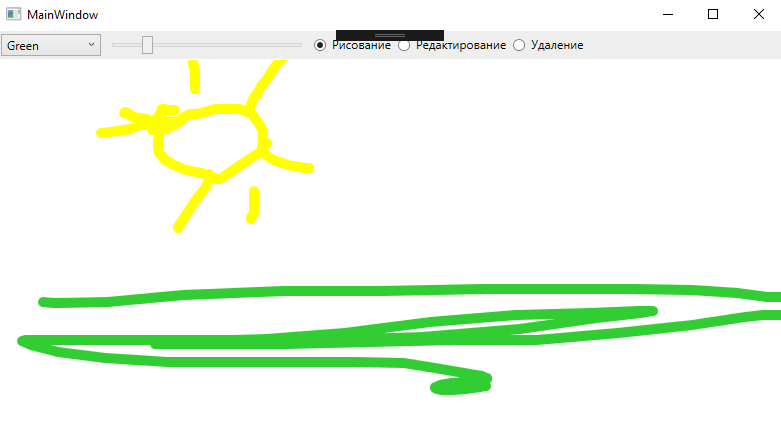


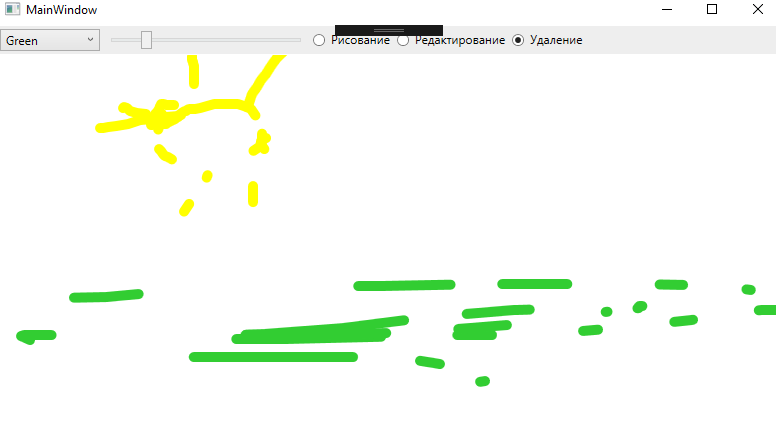




Задание 2: Разработать WPF-приложение «Графический редактор» с выпадающим списком для выбора цвета кисти, ползунком для выбора размеров кисти и зависимыми переключателями для выбора режима работы: «рисование», «редактирование», «удаление».

Работа кода





# Практическая работа № 5

«Привязка данных в графической системе WPF»

Цель работы: Получить практические навыки работы с привязкой данных в графической системе WPF.

Задание 1:

Проверить реакцию среды разработки на неверные значения параметров ElementName и Path. Проанализировать сообщения, которые выводятся в окне вывода (Вид → Вывод) при построении и при запуске приложения.

Код готовой программы:

<Window x:Class="Задание\_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:Задание\_1"

mc:Ignorable="d"

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

<StackPanel>

<Slider Minimum="8" Maximum="30" x:Name="SliderFontSize"></Slider>

<TextBlock x:Name="Message" FontSize="{Binding ElementName=bludov, Path=natyral}">

Пример WPF-приложения для демонстрации привязки данных

</TextBlock>

</StackPanel>

</Window>

Результат готовой программы:



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.

Код готовой программы:

<Window x:Class="Задание\_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:Задание\_2"

mc:Ignorable="d"

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

<StackPanel>

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

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

<Slider x:Name="slider1" />

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

</StackPanel>

</Window>

Результат готовой программы:



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

Код готовой программы:

<Window x:Class="Задание\_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:Задание\_3"

mc:Ignorable="d"

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

<StackPanel>

<Slider Minimum="8" Maximum="30" x:Name="SliderFontSize"></Slider>

<TextBox Text="{Binding ElementName=SliderFontSize, Path=Value}" />

<TextBlock x:Name="Message" FontSize="{Binding ElementName=SliderFontSize, Path=Value}">

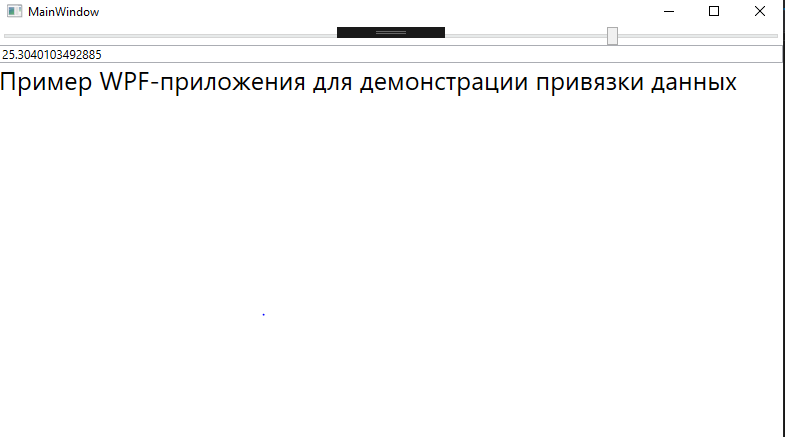
Пример WPF-приложения для демонстрации привязки данных

</TextBlock>

</StackPanel>

</Window>

Результат готовой программы:



# Практическая работа №6

«Использование стилей элементов управления в приложениях WPF»

Цель работы: Ознакомиться со стилями элементов управления WPF-приложений и применить их на практике.

Задание 1:

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

Код программы:

<Window x:Class="Задание\_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:Задание\_1"

mc:Ignorable="d"

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

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Background" Value="Red" />

<Setter Property="Height" Value="50" />

<Setter Property="Margin" Value="10" />

</Style>

</Window.Resources>

<StackPanel Orientation="Horizontal" VerticalAlignment="Top">

<Button Content="Кнопка 1" Background="LimeGreen" />

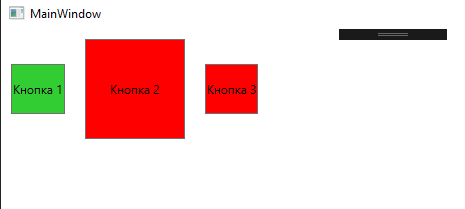
<Button Content="Кнопка 2" Width="100" Height="100" />

<Button Content="Кнопка 3" />

</StackPanel>

</Window>

Результат готовой программы:



Задание 2:

Модифицировать WPF-приложение, разработанное в 3-ей лабораторной работе используя стили для однотипных элементов управления.

Код готовой программы: <Window x:Class="Задание\_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:Задание\_2"

mc:Ignorable="d"

Title="Диспетчеры компоновки" Height="270" Width="692">

<Window.Resources>

<Style TargetType="Label" x:Key="LabelsWrap">

<Setter Property="Margin" Value="10" />

<Setter Property="VerticalContentAlignment" Value="Center" />

<Setter Property="HorizontalContentAlignment" Value="Center" />

<Setter Property="Height" Value="75" />

<Setter Property="Width" Value="65" />

</Style>

<Style TargetType="Button">

<Setter Property="Margin" Value="5" />

</Style>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="140" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

<RowDefinition Height="30"/>

<RowDefinition />

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Left">

<Label Margin="5" Padding="0" VerticalAlignment="Center" Content="Последние документы"/>

<Button Content="Открыть" />

<Button Content="Сохранить" />

<Button Content="Закрыть" />

<Button Content="Обработать" />

<Button Content="О программе" />

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Column="1" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Left">

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

<Label Content="2.txt" />

<Label Content="3.txt" />

</StackPanel>

<StackPanel Grid.Column="0" Grid.Row="1" Grid.RowSpan="2">

<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 Grid.Column="1" Grid.Row="2">

<Label Background="Yellow" Content="A" Style="{StaticResource ResourceKey=LabelsWrap}" />

<Label Background="Red" Content="B" Style="{StaticResource ResourceKey=LabelsWrap}" />

<Label Background="Green" Content="C" Style="{StaticResource ResourceKey=LabelsWrap}" />

<Label Background="Blue" Content="D" Style="{StaticResource ResourceKey=LabelsWrap}" />

<Label Background="Gray" Content="E" Style="{StaticResource ResourceKey=LabelsWrap}" />

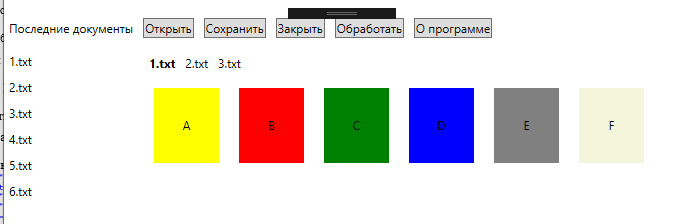
<Label Background="Beige" Content="F" Style="{StaticResource ResourceKey=LabelsWrap}" />

</WrapPanel>

</Grid>

</Window>

Результат готовой программы:



Задание 3:

Разработать 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.

Изменить стиль элемента управления в коде можно следующим образом:

(sender as FrameworkElement).Style =

(Style)Resources["ИМЯ\_СТИЛЯ"];

Код программы: <Window x:Class="Задание\_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:Задание\_3"

mc:Ignorable="d"

Title="MultiEdit" Height="450" Width="800" ResizeMode="CanMinimize">

<Window.Resources>

<Style TargetType="TextBox" x:Key="TextBoxes">

<Setter Property="TextWrapping" Value="Wrap" />

<Setter Property="AcceptsReturn" Value="True" />

<Setter Property="AcceptsTab" Value="True" />

<Setter Property="VerticalScrollBarVisibility" Value="Visible" />

<Setter Property="Margin" Value="8" />

<EventSetter Event="TextChanged" Handler="TextBox\_TextChanged" />

</Style>

<Style TargetType="TextBox" x:Key="Small" BasedOn="{StaticResource ResourceKey=TextBoxes}">

<Setter Property="Height" Value="45" />

<Setter Property="FontSize" Value="12" />

</Style>

<Style TargetType="TextBox" x:Key="Big" BasedOn="{StaticResource ResourceKey=TextBoxes}">

<Setter Property="Height" Value="95" />

<Setter Property="FontSize" Value="24" />

</Style>

</Window.Resources>

<Grid>

<StackPanel Grid.Column="0">

<TextBox Style="{StaticResource ResourceKey=Big}" Name="FirstTextBox1" Tag="Left" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Left" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Left" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Left" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Left" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Left" />

</StackPanel>

<StackPanel Grid.Column="1">

<TextBox Style="{StaticResource ResourceKey=Big}" Name="FirstTextBox2" Tag="Right" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Right" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Right" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Right" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Right" />

<TextBox Style="{StaticResource ResourceKey=Small}" Tag="Right" />

</StackPanel>

<Border BorderBrush="Black" BorderThickness="0,0,0.75,0" Grid.Column="0" />

<Border BorderBrush="Black" BorderThickness="0.75,0,0,0" Grid.Column="1" />

<Grid.Background>

<LinearGradientBrush EndPoint="0,1">

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

<GradientStop Color="FloralWhite" Offset="0.5" />

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

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition />

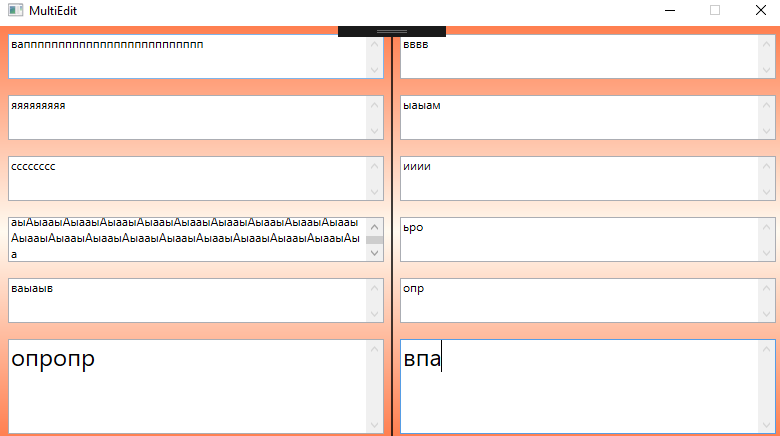
<ColumnDefinition />

</Grid.ColumnDefinitions>

</Grid>

</Window>

Работа программы:



Вывод: В ходе выполнения практической работы было произведено ознакомление со стилями элементов управления WPF-приложений, которые были применены на практике.

Практическая работа №7

«Триггеры в WPF-приложениях»

Цель работы: Ознакомиться с различными триггерами в приложениях WPF.

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.

Код готовой программы:

<Window x:Class="Задание\_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:Задание\_1"

mc:Ignorable="d"

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

<Window.Resources>

<Style TargetType="Button">

<Style.Triggers>

<Trigger Property="IsEnabled" Value="True">

<Trigger.Setters>

<Setter Property="Background" Value="Blue" />

</Trigger.Setters>

</Trigger>

<Trigger Property="IsFocused" Value="True">

<Trigger.Setters>

<Setter Property="Background" Value="Red" />

</Trigger.Setters>

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

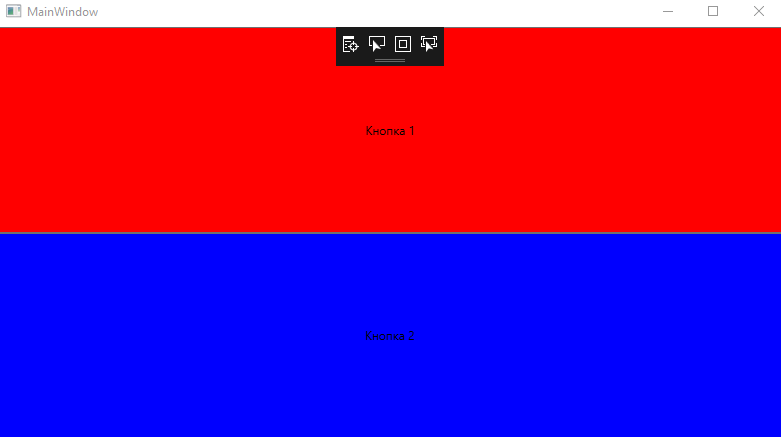
<Button Content="Кнопка 1" Grid.Row="0" />

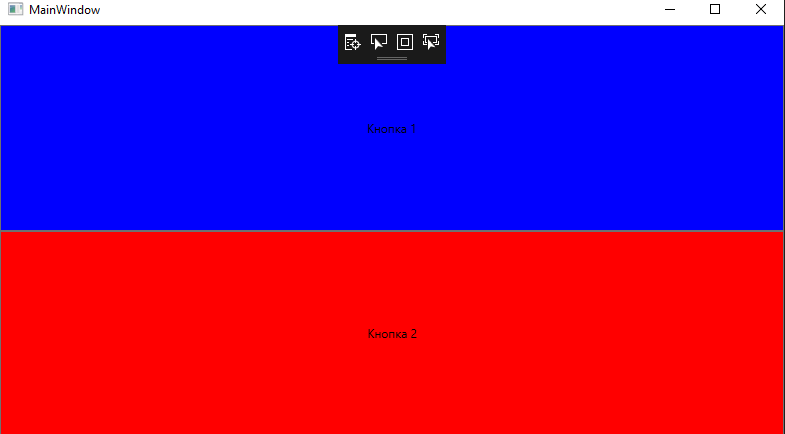
<Button Content="Кнопка 2" Grid.Row="1" />

</Grid>

</Window>

Результат готовой программы:





Задание 2:

Код:

<Window x:Class="Задание\_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:Задание\_2"

mc:Ignorable="d"

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

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Margin" Value="5" />

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="FontSize" Value="17" />

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="FontWeight" Value="Bold" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="TextBox">

<EventSetter Event="TextChanged" Handler="TextBox\_TextChanged" />

<Setter Property="AcceptsReturn" Value="True" />

<Setter Property="TextWrapping" Value="Wrap" />

<Setter Property="VerticalScrollBarVisibility" Value="Visible" />

<Setter Property="Background">

<Setter.Value>

<RadialGradientBrush>

<GradientStop Color="BurlyWood" Offset="0.3" />

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

</RadialGradientBrush>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Border BorderBrush="Black" BorderThickness="0,0,0,1" />

<TextBox Grid.Row="1" Name="TB1" />

<TextBox Grid.Row="2" Name="TB2" />

<StackPanel Orientation="Horizontal" Grid.Row="0">

<Button Content="Открыть" />

<Button Content="Очистить" Click="ClearBtn\_Click" />

<Button Content="Закрыть" Name="CloseBtn" Click="CloseBtn\_Click" />

<ComboBox Margin="5" VerticalContentAlignment="Center" SelectedIndex="0" SelectionChanged="ComboBox\_SelectionChanged">

<ComboBoxItem Content="Стиль 1" FontWeight="Normal" Foreground="Black" FontSize="12" />

<ComboBoxItem Content="Стиль 2" FontWeight="Thin" Foreground="BlueViolet" FontSize="15" />

<ComboBoxItem Content="Стиль 3" FontWeight="DemiBold" Foreground="DarkSeaGreen" FontSize="10" />

<ComboBoxItem Content="Стиль 4" FontWeight="ExtraBold" Foreground="Yellow" FontSize="12" />

<ComboBoxItem Content="Стиль 5" FontWeight="Regular" Foreground="Crimson" FontSize="14" />

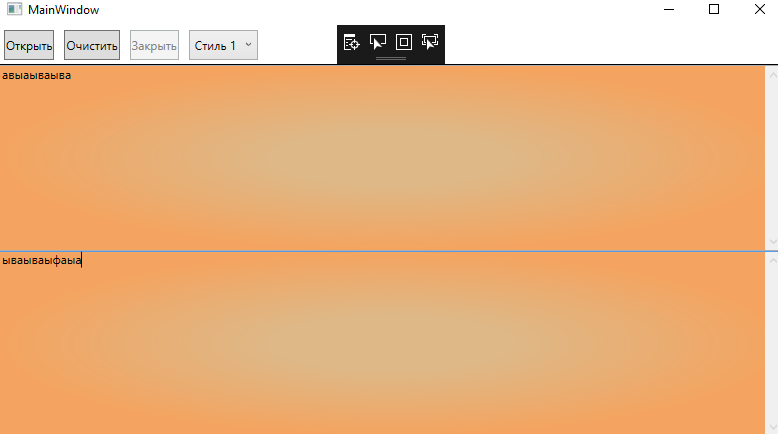
</ComboBox>

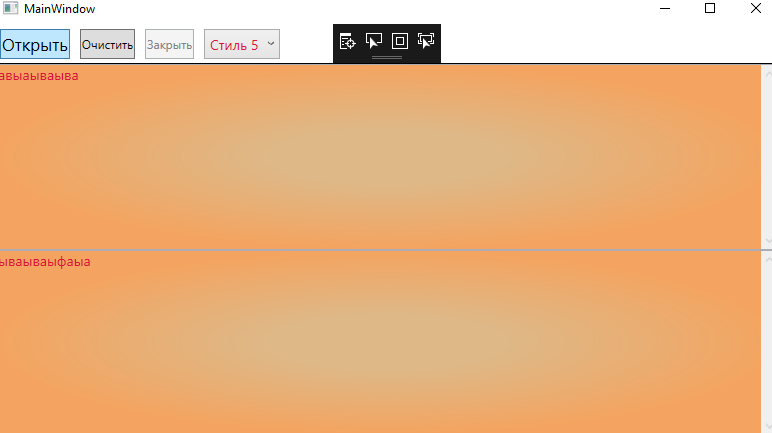
</StackPanel>

</Grid>

</Window>

Работа программы:





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: В ходе выполнения практической работы было произведено ознакомление с принципом работы различных триггеров в приложениях WPF.

# Практическая работа № 8

«Свойства элементов управления. Использование кистей в WPF-приложениях.»

Цель работы: Ознакомиться с различными элементами управления и принципами работы кистей. Разработать приложение WPF с анимацией.

Задание 1:

Код готовой программы:

<Window x:Class="Задание.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:Задание"

mc:Ignorable="d"

Title="Шар" Height="500" Width="500" ResizeMode="CanMinimize">

<Grid>

<StackPanel Margin="0,0,0,50" VerticalAlignment="Bottom" HorizontalAlignment="Center">

<Ellipse Height="200" Width="200" Name="shar" Panel.ZIndex="10">

<Ellipse.Fill>

<RadialGradientBrush x:Name="GrBrush" GradientOrigin="0.3,0.3">

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

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

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

<Rectangle Panel.ZIndex="5" Height="100" Width="{Binding ElementName=shar, Path=Width}" Opacity="0.35" RenderTransformOrigin="0,0.5" >

<Rectangle.Fill>

<VisualBrush Visual="{Binding ElementName=shar}" />

</Rectangle.Fill>

<Rectangle.RenderTransform>

<ScaleTransform ScaleY="-1"></ScaleTransform>

</Rectangle.RenderTransform>

</Rectangle>

</StackPanel>

<Canvas>

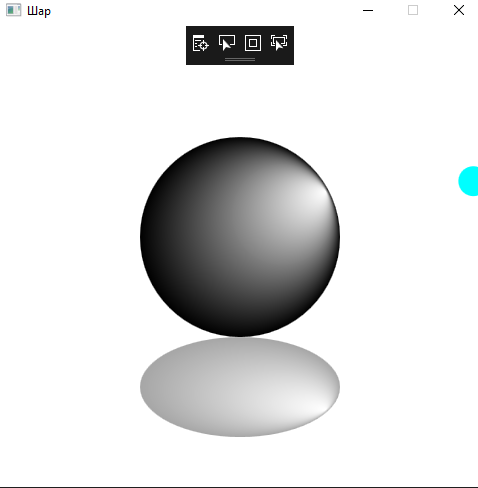
<Ellipse Width="30" Height="30" Fill="#00FFFF" Name="light" Canvas.Left="150" Canvas.Top="150" />

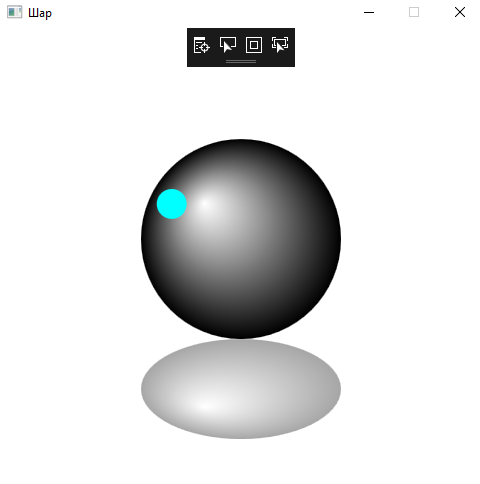
</Canvas>

</Grid>

</Window>

Результат готовой программы:





Вывод: В ходе выполнения практической работы было произведено ознакомление с различными элементами управления и принципами работы кистей, а +также разработано собственное WPF-приложение с анимацией источника света, светового пятна от него на шаре и отражения шара.