

МИНОБРАЗОВАНИЯ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет Московский институт электронной техники»

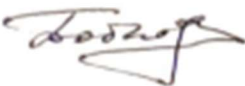
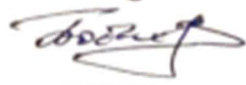
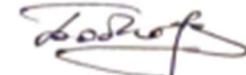
Институт Микроприборов и систем управления

НАПРАВЛЕНИЕ: 27.03.04 «Управление в технических системах»

ДИСЦИПЛИНА: Объектно-ориентированное программирование

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Графический интерфейс пользователя с WPF»

Допущен 20.10.22 
Выполнен 20.10.22 
Защитен 20.10.22 

Работу выполнил
студент гр. УТС-21



(подпись студента)

Р. В. Закшевский

(Ф.И.О. студента)

Преподаватель



(подпись преподавателя)

В. Д. Бобков

(Ф.И.О. преподавателя)

Бобков

Москва, 2022 г.

Цель работы: знакомство со средой разработки Visual Studio на языке программирования C#, принципами работы с WPF приложением.

Задача работы: в данной лабораторной работе требуется создать приложение с графическим интерфейсом пользователя (GUI) используя набор библиотек WPF.

Теоретическая часть.

Новый проект

Для создания проекта с использованием WPF, необходимо открыть диалог “New Project” и выбрать тип шаблона – “Windows Forms Application”. Затем задать корректное имя проекта, и нажать «ОК».

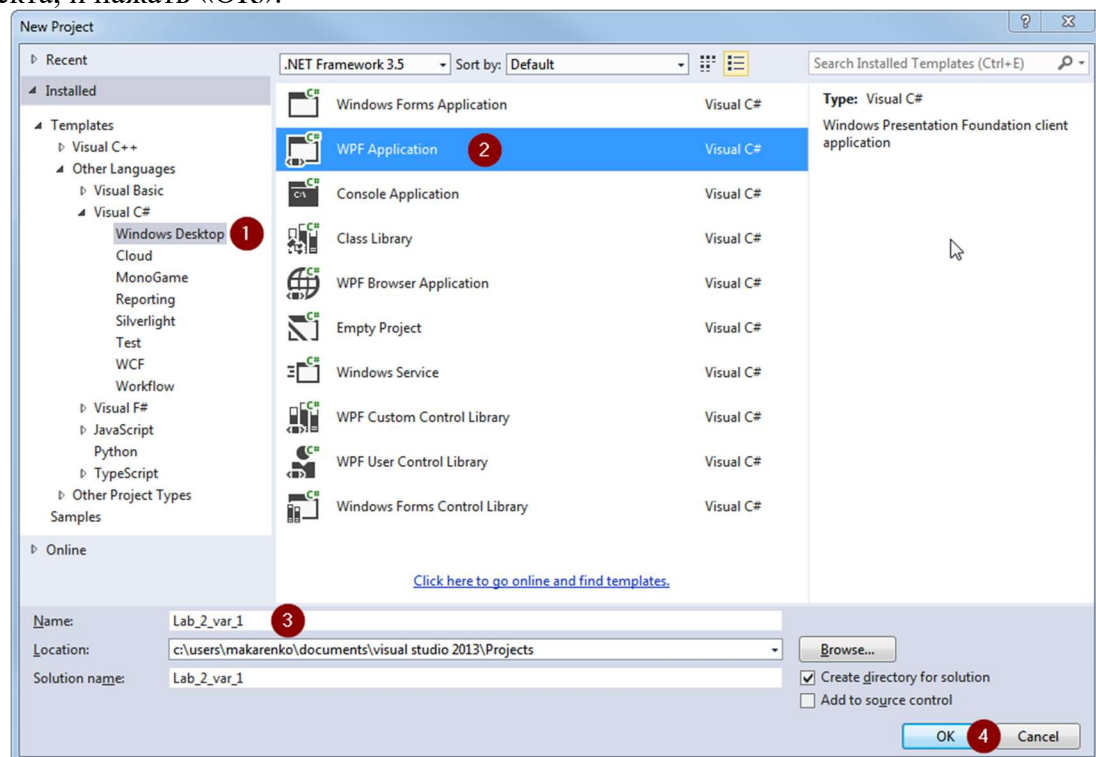


Рисунок 1

После этого перед вами откроется окно дизайнера форм, в котором будет открыта ваша первая форма приложения.

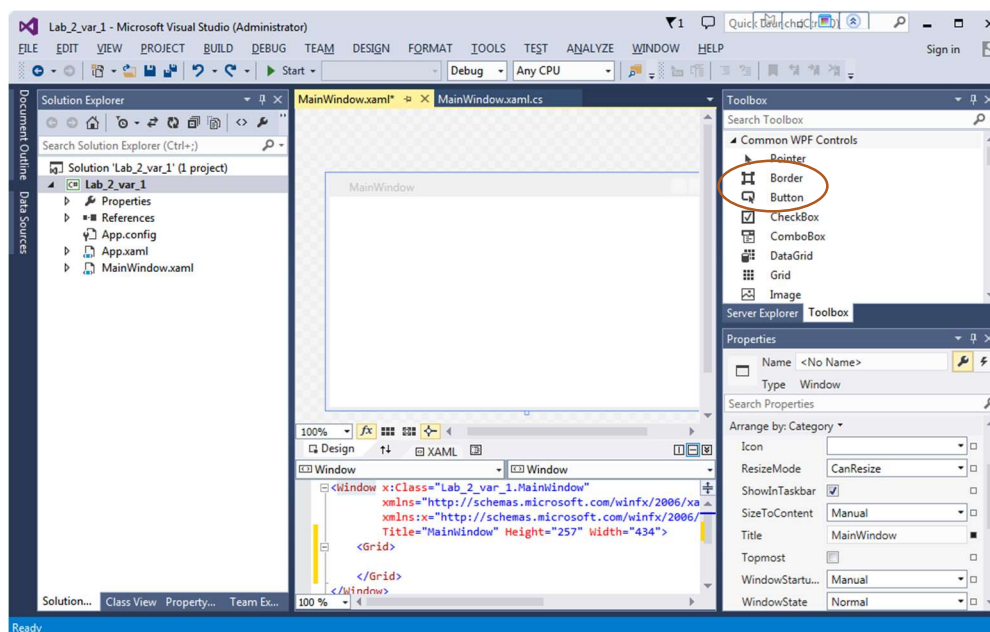


Рисунок 2

По умолчанию форма пустая и не содержит никаких элементов управления, чтобы их добавить следует воспользоваться панелью **Toolbox**, открыть которую можно нажатием на соответствующую кнопку в правой части окна. Чтобы панель не пропадала после потери фокуса, ее можно закрепить нажатием на иконку с изображением булавки заголовке.

Графические элементы

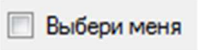
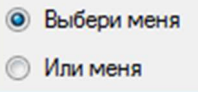
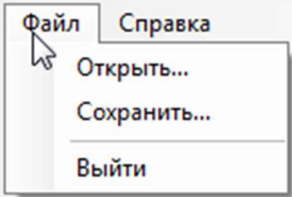
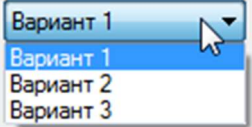
В открывшейся панели **Toolbox** содержатся стандартные графические элементы: кнопки, поля ввода, тестовые подписи, которые вы можете использовать в своем приложении. Подробное описание доступных элементов можно найти в справочнике разработчика – MSDN, для этого нужно нажать на интересующий элемент и нажать клавишу F1, после чего в браузере откроется страница с описанием элемента.

Чтобы добавить выбранный элемент на форму, нужно либо дважды кликнуть по нему, либо перетащить из **Toolbox** на форму.

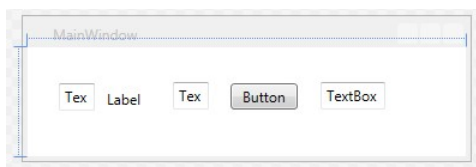
Наиболее часто используемые элементы:

Таблица 1

Название	Изображение	Описание
Button		Обычная кнопка
TextBox		Поле ввода, используется для ввода текста с клавиатуры. Может быть однострочным и многострочным
Label		Тестовая метка, используется для подписи полей ввода, списков, а так же

		для вывода небольших информационных сообщений
CheckBox		Флажок, используется для включения и выключения каких-либо настроек
RadioButton		Переключатель, используется для выбора из нескольких вариантов
Menu		Главное меню программы
ComboBox		Выпадающий список

Добавим на форму несколько элементов: три поля ввода, одну кнопку и одну текстовую метку:



Свойства элементов

После этого, необходимо настроить свойства каждого элемента. Для этого необходимо открыть и зафиксировать панель “Properties”. В открывшейся панели будут отображаться все свойства активного (выбранного) элемента.

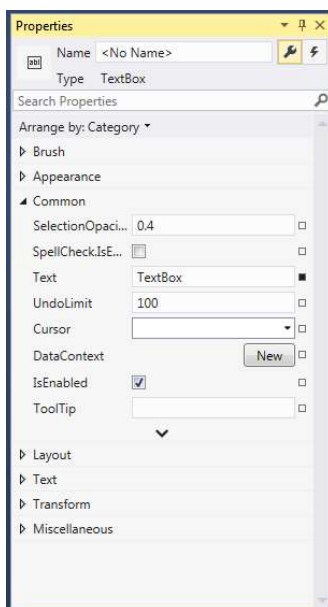


Рисунок 3

Подробное описание каждого свойства можно узнать в MSDN (выделить свойство и нажать F1). Здесь приведены только некоторые, наиболее важные свойства.

Таблица 2

Название	Описание
Name	Название элемента. Именно по этому имени к элементу можно будет обращаться из кода. Ограничения такие же, как и на именованние переменных в коде. Необходимо всегда задавать корректное и осмысленное значение поля Name!
Content или Text	Текст, отображающийся в элементе.
IsEnabled	«Включенность» элемента. Это свойство включает или выключает возможность взаимодействия с элементом (нажатие на кнопку, ввод текста в поле, переключение флага)
Visibility	Видимость элемента. Позволяет временно скрывать элемент в соответствии с алгоритмом программы
Brush	Цвет текста\фона элемента
AcceptsReturn (только TextBox)	Вывод текста в многострочном режиме
IsReadOnly (только TextBox)	Запрещает редактировать текст в поле ввода
Items (только ComboBox)	Элементы списка

Зададим нужные свойства.

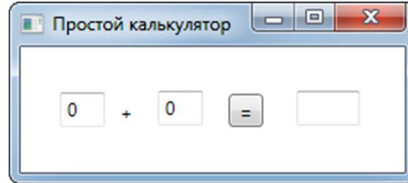


Рисунок 4

В первую очередь необходимо задать имена элементов, имена нужно подбирать так, чтобы, перейдя к написанию кода, можно было легко вспомнить, как каждый из элементов называется. Рекомендуется использовать «венгерскую» нотацию для именования графических элементов, то есть смысловой части имени должно предшествовать двух-трехбуквенное сокращенное название типа элемента. Например, TextBox – tb, Button – btn, Label – lbl и так далее. Это может пригодиться, в тех случаях, когда вы помните тип элемента, но не помните точно его названия, тогда после ввода первых букв типа, автодополнение редактора подскажет вам все элементы указанного типа. Зададим для добавленных элементов следующие имена tbA, tbB – для полей ввода первого и второго операндов, lblPlus – для символа +, btnCalculate – для кнопки, по которой будет запускаться вычисление и tbResult для поля результата.

Для поля результата нужно выставить свойство ReadOnly в true, так как пользователь не должен изменять его значение.

Для формы, кнопки и текстовой метки зададим соответствующие значения Text.

Обработчики событий

Если запустить программу сейчас, то она запустится, кнопка будет нажиматься, в поля ввода можно будет вводить числа, но никаких вычислений происходить не будет, поскольку мы только создали интерфейс программы, но не написали код, который бы заставил программу делать то, что мы от нее хотим. Теперь нужно организовать взаимодействие с пользователем.

Действия, которые производит пользователь с программой, называются **событиями (events)**. Событиями могут быть щелчок кнопкой мыши, перемещение мыши, и множество других. Функции, которые вызываются при наступлении события, называются **обработчиками событий (event handlers)**. При разработке пользовательского интерфейса основная задача, это создание обработчиков событий.

В нашей программе основное событие – это нажатие пользователем на кнопку «=», поэтому необходимо добавить собственный обработчик этого нажатия. Для этого достаточно дважды кликнуть по кнопке на форме. После чего откроется окно с кодом, в котором среда автоматически сгенерирует шаблон обработчика события:

```
private void btnCalculate_Click(object sender, RoutedEventArgs e)
{
}
}
```

По своей сути, обработчик события – это обычный метод класса формы, который принимает два параметра:

- object sender – ссылка на объект который отправил событие (в нашем случае это btnCalculate)

- RoutedEventArgs e – описание произошедшего события

Так как это обычный метод, то в нем можно писать обычный код:

```
private void btnCalculate_Click(object sender, EventArgs e)
{
    int a = int.Parse(tbA.Text);
    int b = int.Parse(tbB.Text);
    int result = a + b;
    tbResult.Text = String.Format("{0}", result);
}
```

- Первая строка: считать значение, вписанное в первое текстовое поле, преобразовать его в число и сохранить в переменной **a**.
- Вторая строка: считать значение, вписанное во второе текстовое поле, преобразовать его в число и сохранить в переменной **b**.
- Третья строка: сложить два числа и поместить в переменную **result**.
- Четвертая строка: преобразовать результат в строку и вывести в текстовое поле **tbResult**.

Тут видно, что ко всем графическим элементам можно обращаться из кода по тому имени, что задано в поле Name в панели свойств. Так же, из кода, доступны все свойства объекта, к ним можно обращаться через точку после имени объекта.

После этого можно запустить программу и проверить ее в действии:

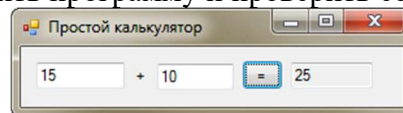


Рисунок 5

Усложняем задачу

Теперь попробуем добавить возможность не только складывать, но и вычитать, умножать и делить.

Для этого добавим на форму ComboBox, который будет определять операцию которую нужно произвести. В этот ComboBox добавим 4 элемента (свойство Items, тип ComboBoxItem) – «+», «-», «*», «/».

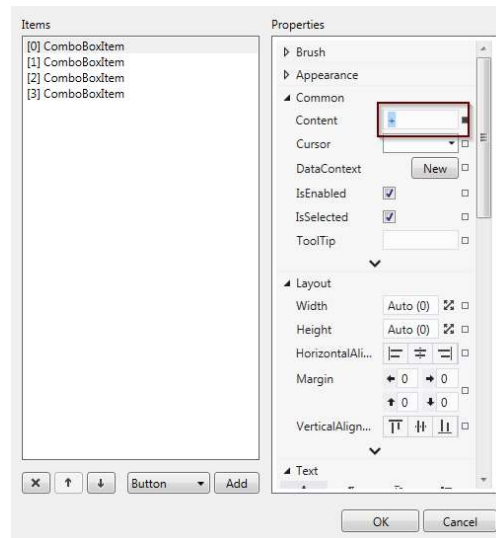


Рисунок 6

После этого форма должна выглядеть примерно так:

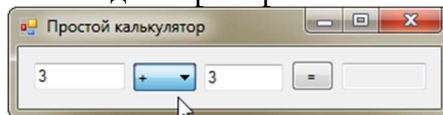


Рисунок 7

И перепишем обработчик кнопки следующим образом:

```
private void btnCalculate_Click(object sender, RoutedEventArgs e)
{
    int a = int.Parse(tbA.Text);
    int b = int.Parse(tbB.Text);

    int operation = cbOperation.SelectedIndex;
    int result = 0;

    switch (operation)
    {
        case 0:
            result = a + b;
            break;
        case 1:
            result = a - b;
            break;
        case 2:
            result = a * b;
            break;
        case 3:
            result = a / b;
            break;
    }
    tbResult.Text = String.Format("{0}", result);
}
```

Первая часть кода осталась неизменной, разберем вторую половину.
`int operation = cbOperation.SelectedIndex;`

Свойство `SelectedIndex` у класса `ComboBox` содержит индекс текущего выбранного элемента. Если вы добавили элементы в том порядке, что указано выше, то индексы будут соответствовать следующим операциям:

0. Сложение
1. Вычитание
2. Умножение
3. Деление

Таким образом, в переменной `operation` сохраняется код выбранной операции. Затем, используя этот код, в операторе `switch` мы выбираем, какое из действий необходимо произвести над операндами. После чего выводим результат в поле результата.

Так как пользователь может ввести второй аргумент равный 0, то может возникнуть ситуация деления на ноль, и ее нужно обработать. Для этого добавим следующие строки в часть отвечающую за деление:

```
case 3:
    if (b == 0)
    {
        MessageBox.Show("На ноль делить нельзя!");
        break;
    }
    result = a / b;
    break;
```

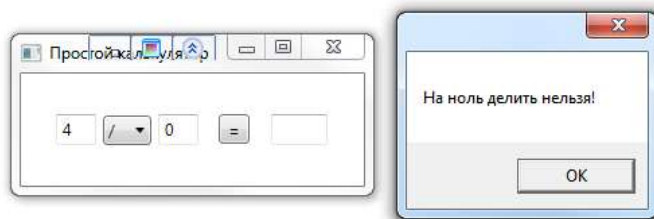


Рисунок 8

Таким образом, с помощью объекта `MessageBox` можно показывать различные уведомления для пользователя. Вообще, для обработки таких ситуаций существует отдельный механизм, называющийся `Exceptions`.

Задание.

Вариант 5

Создать программу, которая позволит раскрашивать цвет формы в различные цвета (свойство `Background`). Предоставить возможность выбрать цвет из списка (не менее 10 различных цветов), либо задав цвет при помощи значений красного, зеленого и синего (RGB). Для задания цвета по значениям используйте класс `SolidColorBrush`, например.

Код программы:

```

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 Lab2_Zakshevskij_UTS_22
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void Bt_Click(object sender, RoutedEventArgs e)
        {
            if (Rb1.IsChecked==true)
            {
                int ColorIndex = Cb1.SelectedIndex;
                switch (ColorIndex)
                {
                    case 0:
                        Wind.Background = Brushes.Red;
                        break;
                    case 1:
                        Wind.Background = Brushes.Orange;
                        break;
                    case 2:
                        Wind.Background = Brushes.Yellow;
                        break;
                    case 3:
                        Wind.Background = Brushes.Green;
                        break;
                    case 4:
                        Wind.Background = Brushes.Blue;
                        break;
                }
            }
            if (Rb2.IsChecked==true)
            {
                string A = Tb1.Text;
                string F = Tb2.Text;
                string C = Tb3.Text;
                int R = 0;
                int G = 0;
                int B = 0;
                bool res;
            }
        }
    }
}

```

```

        if ((int.TryParse(A, out R)) && (int.TryParse(F, out
G)) && (int.TryParse(C, out B)))
        {
            res = true;
            R = int.Parse(A);
            G = int.Parse(F);
            B = int.Parse(C);
        }
        else
        {
            res = false;
            MessageBox.Show("Данные введены некорректно! Повторите ввод.");
        }
        if (res == true)
        {
            Wind.Background = new SolidColorBrush(Color.FromArgb(255,
(byte)R, (byte)G, (byte)B));
        }
    }
}

```

Внешний вид программы:

Закшевский Родион, УТС-22. Вариант 5

☒ Из списка

☐ По значениям

R: 0

G: 0

B: 0

Сменить цвет

Рисунок 9 Внешний вид приложения

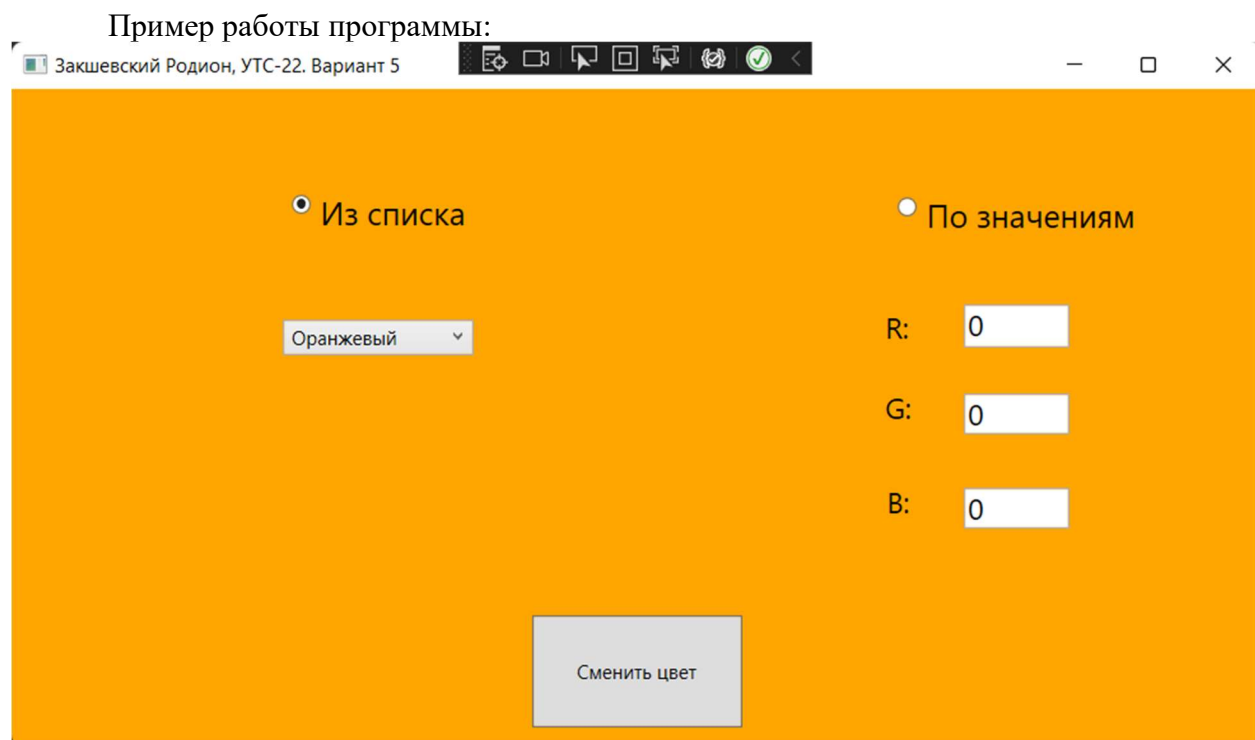


Рисунок 10 Пример работы программы

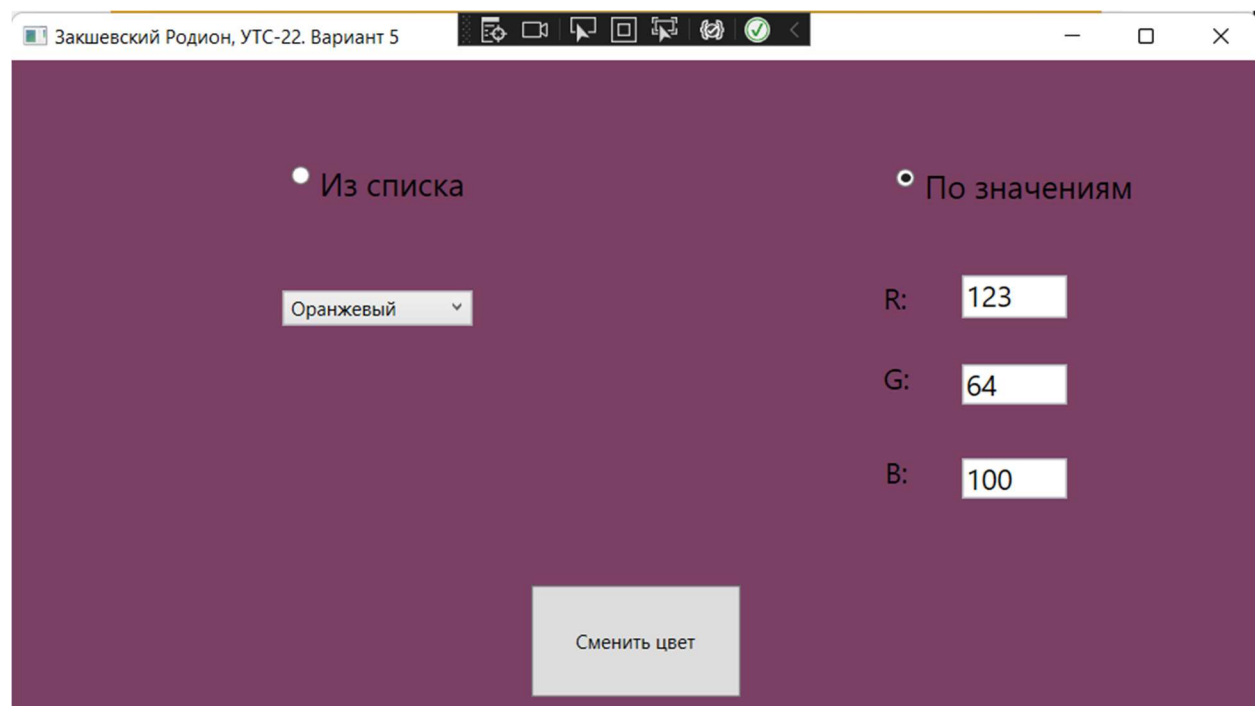


Рисунок 11 Пример работы программы 2

Вывод: в данной работе мы ознакомились со средой разработки Visual Studio на языке программирования C#, принципами работы с WPF приложением, сделали простое оконное приложение.