

Рязанский станкостроительный колледж РГРТУ

Основы алгоритмизации и программирования

Тема 8. Визуальное проектирование.

Рязань 2022

Оглавление

Общие сведения	3
Объектно-ориентированное программирование	3
Визуальное проектирование	3
Принципы разработки Windows – приложений	3
Создание программы Windows с применением WPF .NET	4
Основные сведения	4
Создание проекта приложения WPF.....	4
Правила создания проекта:	6
Структура проекта	7
Конструктор формы (MainWindows.xaml)	7
Редактора кода (MainWindows.xaml.cs)	7
Обозреватель решений	8
Свойства	9
Оформление кода программы в соответствии со стандартом кодирования	9
Основные положения	9
Правила именования элементов управления (кнопки, текстовые поля и т.д.).....	9
Правила именования событий	10
Создание программы Windows с использованием среды Visual Studio при визуальном проектировании	11
Алгоритм разработки программы	11
Типовой алгоритм разработки обработчика событий.....	11
Пример разработки программы сложения 2-х чисел	11
Тюнинг приложения для создания удобной и отказоустойчивой программы.....	15
Практическая работа №17	17

Общие сведения

Объектно-ориентированное программирование

Объектно-ориентированное программирование (сокращенно ООП) является методом программирования, имитирующим то, как человек выполняет какую-либо работу. В основе ООП лежит понятие объект.

Объект – это некоторая структура, соответствующая объекту реального мира, его поведению. Программа в таком случае представляет собой набор объектов и их взаимосвязей.

Объектами, используемыми при проектировании программы с визуальным интерфейсом, являются: окна, кнопки, поля для ввода информации и т.д.

Для правильного использования объектов нужно четко понимать основное свойство ООП – инкапсуляцию.

Инкапсуляция - это комбинирование данных с функциями, которые манипулируют этими данными, для получения нового типа данных – объекта, а также защита данных от внешнего вмешательства и неправильного использования. Связанные с объектом функции называются **методами**. Данные – называются **полями или свойствами**.

Таким образом, для использования объектов при проектировании интерфейса необходимо изучить и знать свойства и методы этих объектов.

Визуальное проектирование

Визуальное проектирование — это способ создания программы путём манипулирования графическими объектами вместо написания её текста.

При таком подходе среда разработки обладает библиотекой разработанных визуальных элементов, которые используются для быстрого проектирования интерфейса программы. Интерфейс программы создаётся путем простого перетаскивания мышкой нужного элемента из библиотеки в окно проектируемой программы и его настройки.

Принципы разработки Windows – приложений

Большинство современных программ имеют оконный интерфейс. Это значит, что их внешний вид и поведение соответствует стандартам современных операционных систем. Это поведение отличается от консольных программ и требует иного подхода к разработке логики работы программы.

Логика работы консольной программы:

1. Выполнение программы начинается, как правило, с вывода на экран консольного окна.
2. Далее программа выполняет последовательно все операторы. Приостановка программы происходит только для операторов ввода, для ввода значений с клавиатуры.
3. Завершение работы программы происходит после выполнения последнего оператора и консольное окно закрывается.

Логика работы оконной программы:

1. Выполнение программы начинается с вывода на экран главного окна.
2. Далее программа выполняется обычно в режиме диалога с пользователем. Это означает, что если пользователь не осуществляет каких-либо управляющих воздействий на программу (например, с помощью мыши или клавиатуры), то эта программа находится в режиме ожидания.
3. Активные действия программы проявляются в виде реакции на **управляющие воздействия (события)**.
4. Завершение работы программы обычно также происходит по инициативе пользователя и приводит к закрытию окна.

Как мы видим основной принцип разработки оконных программ – это **реакция программы на действия пользователя**. Для этого почти каждый визуальный элемент имеет список событий, на которые он может реагировать, и задача программиста состоит в том, чтобы писать программный код реализации этих событий.

Создание программы Windows с применением WPF .NET

Основные сведения

Для создания программ в среде *Windows* используются два подхода реализации интерфейса, использование *Windows Forms* или использование *WPF (Windows Presentation Foundation)*.

Последнее время для разработки интерфейса все чаще используют *WPF*, который был разработан позже, чем *Windows Forms* и обладает рядом преимуществ:

1. Декларативное определение графического интерфейса с помощью специального языка разметки *XAML*, основанном на *xml*.
2. Независимость от разрешения экрана: поскольку в *WPF* все элементы измеряются в независимых от устройства единицах, приложения на *WPF* легко масштабируются под разные экраны с разным разрешением.
3. Новые возможности, которых сложно было достичь в *WinForms*, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др.
4. Богатые возможности по созданию различных приложений: это и мультимедиа, и двумерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимации.
5. Аппаратное ускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной.

Создание проекта приложения WPF

1. В стартовом окне выбрать команду создание проекта или в окне среды разработки Visual Studio выбрать команду **Файл – Создать – Проект**.
2. Появится окно выбора типа проекта см. рисунок 1, в котором выбрать элемент **Приложение WPF (Майкрософт) Проект WPF .NET** и нажать кнопку **Далее**.

Замечание: Так как список элементов может быть большой и найти нужный элемент не всегда может быть просто, используйте функции фильтра. Фильтр см. рисунок 4 имеет 4 элемента:

- 1) Строка поиска по названию проекта - wpf.
- 2) Выбор языка проекта – это с#
- 3) Выбор платформы – Windows.
- 4) Выбор типа проекта – Рабочий стол.

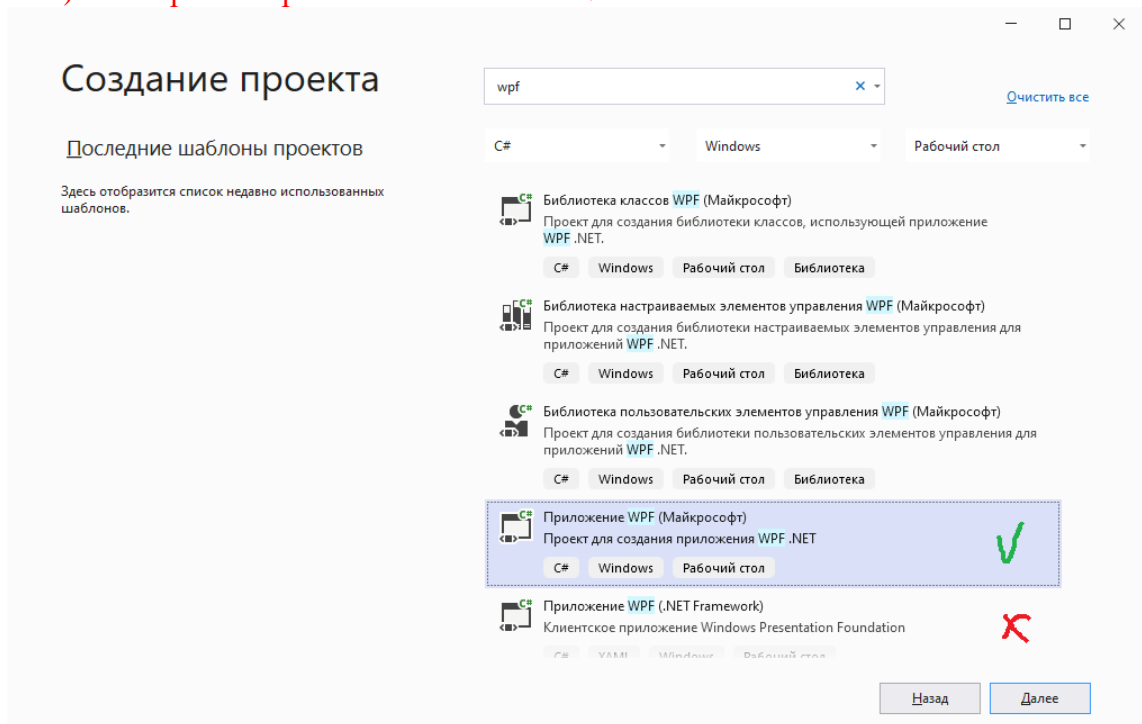


Рисунок 1 - Окно создания проекта

3. Укажите имя, расположение файлов проекта программы, см. рисунок 2 и нажмите кнопку *Далее*.

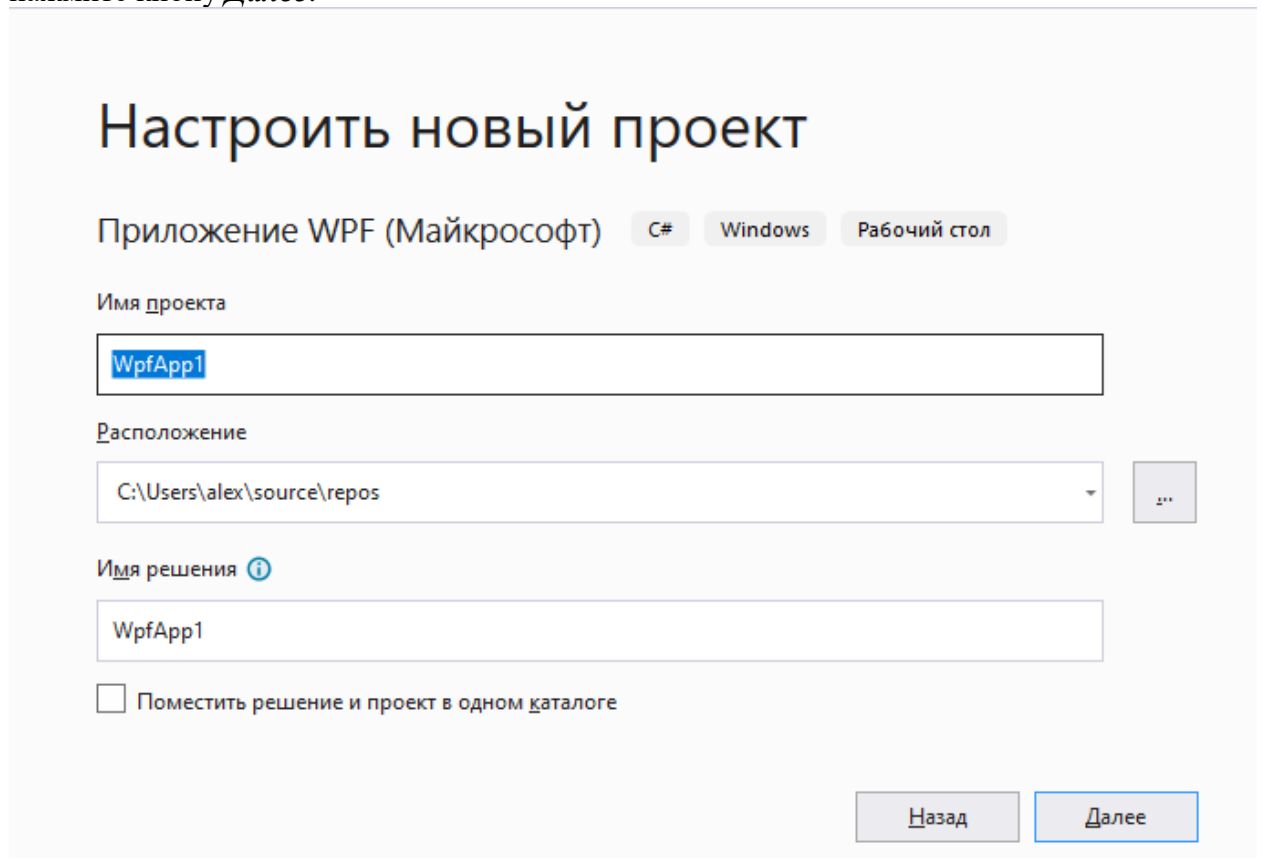


Рисунок 2 - Выбор имени и месторасположение файлов проекта

4. Укажите версию платформы *.Net* см. рисунок 3 и нажмите кнопку *Создать*.

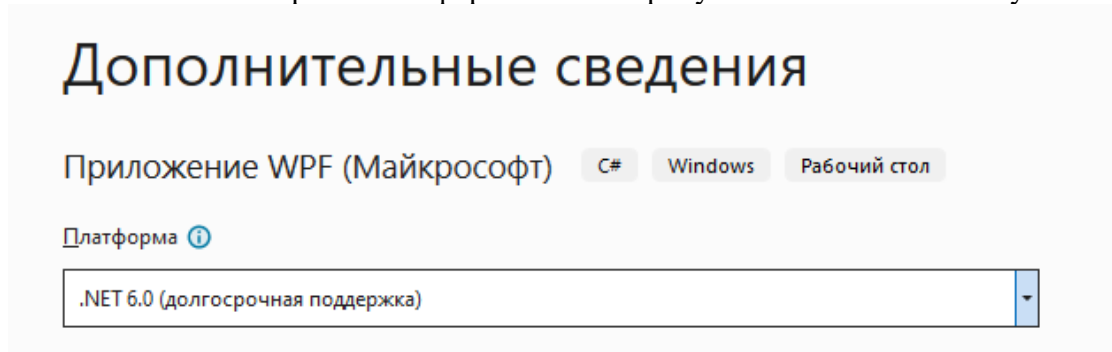


Рисунок 3 - Выбор версии платформы *.Net*

Правила создания проекта:

1. Желательно не использовать стандартных названий вроде WpfApp1, название проекта должно отображать его содержимое, например, Лаба1.
2. Обязательно сохранять каждую программу в отдельную папку. Это позволит легко найти нужную программу и не запутаться в файлах, входящих в данную программу.

При создании проекта программы *Windows WPF* появится заготовка программы, которая имеет следующий вид.

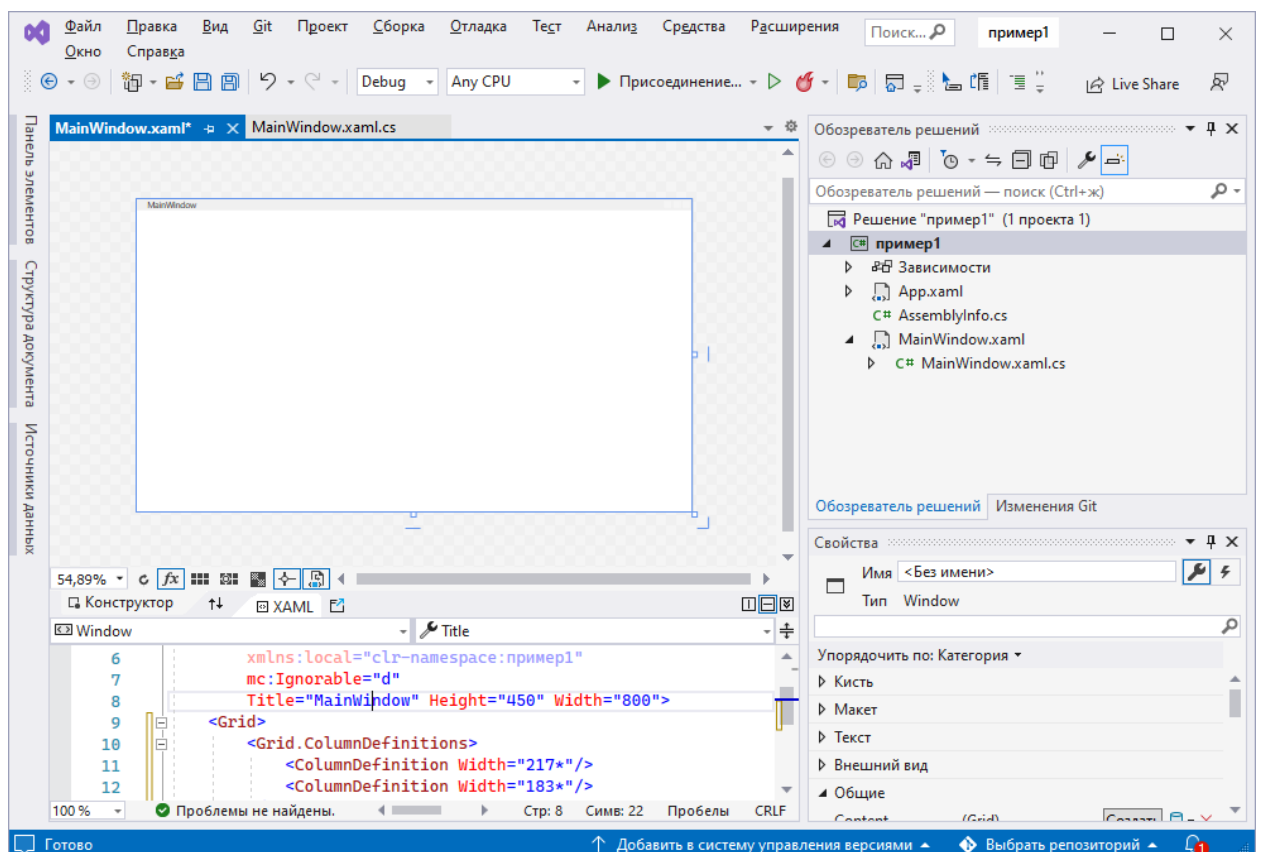


Рисунок 4 - Заготовка программы Windows WPF

Структура проекта

Конструктор формы (MainWindows.xaml)

Представляет собой окно, обычно разделенное на две части. Верхняя часть окна содержит визуальное представление интерфейса. Нижняя часть окна содержит разметку интерфейса в *XAML*.

Верхняя часть окна представляет обычную форму, которая является типичным окном любой программы. На форме размещаются другие визуальные элементы интерфейса программы.


Элементы берутся из окна «*Панель элементов*», которая обычно скрыта и находится в левой панели среды разработки. Размещение визуальных элементов в проектируемом окне программы осуществляется обычным перетаскиванием элемента из окна «*Панель элементов*» в проектируемое окно программы, при этом в окне с разметкой интерфейса *XAML* автоматически появляется код разметки этого элемента.

При отсутствии окна «*Панель элементов*» включите его командой «*Вид – Панель элементов*».

В нижней части окна располагается разметка интерфейса в *XAML*. Разметка формируется автоматически при добавлении элемента в окно программы. Также формировать интерфейс можно вручную, добавляя соответствующие элементы в разметку интерфейса и указывая их атрибуты (свойства).

Как правило, элементы добавляют с помощью визуального проектирования, а настройка и изменение атрибутов происходит в разметке интерфейса в *XAML*.

На рисунке 5 можно видеть элемент *Window*, который представляет окно программы. Элемент *Window* содержит атрибуты: *Title* – заголовок окна, *Height* – высота окна, *Width* – ширина окна. Элемент *Window* содержит вложенный элемент *Grid*, который является *контейнером для размещения других элементов*. В WPF все элементы должны размещаться в контейнерах, о чем подробнее рассмотрим позднее.



```
<Window x:Class="пример1.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:пример1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
    </Grid>
</Window>
```

Рисунок 5 - Заготовка разметки интерфейса в *XAML*

Редактора кода (MainWindows.xaml.cs)

Это окно является программным редактором и используется для написания кода программы.

В данном окне содержится заготовка класса окна нашей будущей программы. Перед классом с помощью директивы *using* подключаются необходимые библиотеки, тех визуальных компонентов, которые, будут использоваться в программе.

На рисунке 6 отображена заготовка программного кода программы *Windows*, которую не стираем и ничего в нее не добавляем.

```

using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace пример1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    Ссылка 2
    public partial class MainWindow : Window
    {
        Ссылка 0
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}

```

Рисунок 6 - Заготовка кода программы Windows WPF

Обозреватель решений

Содержит список основных элементов, используемых в решении или проекте, и показывает структуру этих элементов.

На рисунке 7 располагается элемент, содержащий разметку окна программы (*MainWindows.xaml*). Двойной щелчок на узел «*MainWindows.xaml*» открывает конструктор формы. При выделенном элементе «*MainWindows.xaml*» нажатие на клавишу **F7** открывает код класса формы, который содержит обработчики событий (программный код) формы или её элементов.

Элемент (*MainWindows.xaml.cs*) содержит код класса формы, который содержит обработчики событий (программный код) формы или её элементов.

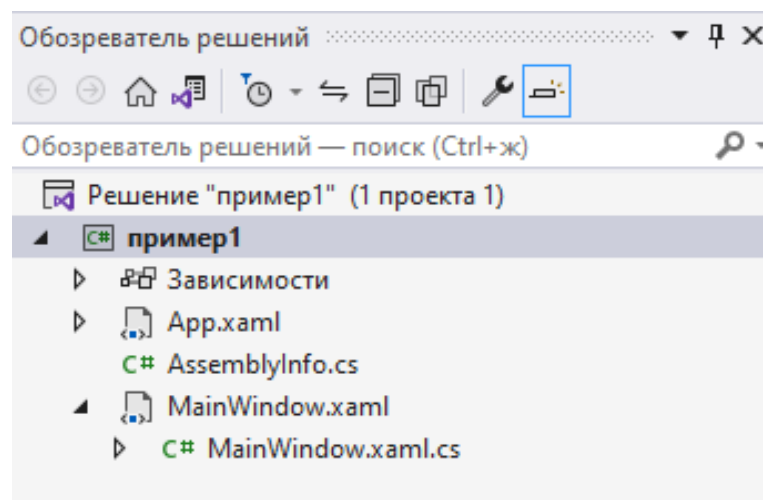


Рисунок 7 - Выбор и открытие элементов проекта

Свойства

Окно свойств, которое показывает свойства (*Properties*) и события (*Events*) выбранного элемента. Оно обеспечивает простой и удобный интерфейс для изменения свойства и управления событиями, на которые реагирует элемент.



- кнопка работы со свойствами.



- кнопка работы с событиями.

На рисунке 8 мы видим свойства элемента *MainWindow* и выделенное свойство *Title*, которое позволяет задать заголовок окна.

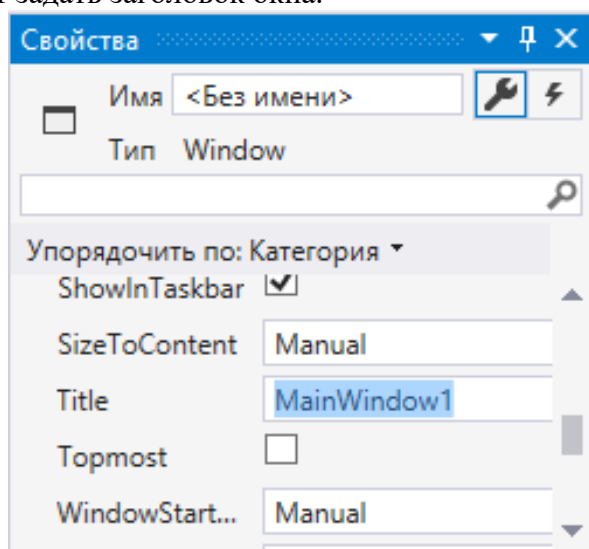


Рисунок 8 - Окно свойств выделенного элемента

Оформление кода программы в соответствии со стандартом кодирования

Основные положения

Стандарт является соглашением по оформлению и написанию кода на языке C#. В документе приведены основные правила оформления кода и приемы, используемые при написании программ.

Стандарт является обязательным для любых разработок программных систем, а также разработки учебных программ в рамках учебного процесса РССК РГРТУ.

Далее мы рассмотрим основные положения по оформлению кода на языке C#, более подробно смотрите файл «*Стандарт оформления кода C#*».

Правила именования элементов управления (кнопки, текстовые поля и т.д.)

Для элементов управления используется стиль CamesCase.

Имя элемента должно быть отражать назначение элемента управления, начинается имя с префикса типа элемента.

Пример: btnOpenFile, tbLogin

Список префиксов:

Button – btn,

CheckBox – cb,
Label – l,
ListBox – lb,
Panel – pnl
RadioButton – rb,
TextBox – tb,
DataGrid – dg,
RadioButtonList – rbl.

Правила именования событий

Для событий используется стиль CamesCase.

Имя события должно содержать наименование элемента и наименование события этого элемента соединённые через нижнее подчеркивание.

Пример: btnOpenFile_Click, tbLogin_TextChanged

Создание программы Windows с использованием среды Visual Studio при визуальном проектировании

Алгоритм разработки программы

1. Внимательно читаем задание, определяем входные и выходные данные.
2. Разрабатываем (ищем) методы решения задачи.
3. Разрабатываем алгоритм решения задачи.
4. Проектируем и настраиваем интерфейс программы: размещаем элементы для ввода исходных данных (обычно поля ввода **TextBox**), размещаем элементы для вывода ответа (обычно поля вывода **TextBox**), размещаем управляющие элементы (обычно кнопки **Button**), размещаем поясняющие подписи (обычно **Label**), настраиваем элементы интерфейса.
5. Продумываем логику работу программы - это события, на какие будет реагировать наша программы. Чаще всего это нажатие на кнопку **Рассчитать**. В более сложном случае взаимодействие событий и элементов программы.
6. Разрабатываем обработчики событий, т.е. пишем программой код.

Типовой алгоритм разработки обработчика событий

1. Получаем с формы исходные данные.

```
int x1, x2, z;  
x1 = Convert.ToInt32(textBox1.Text);  
x2 = Convert.ToInt32(textBox2.Text);
```
2. Рассчитываем по решению по условию задачи.

```
z = x1 + x2;
```
3. Выводим ответ на форму.

```
textBox3.Text = Convert.ToString(z);
```

Пример разработки программы сложения 2-х чисел

1. Внимательно читаем задание, определяем входные и выходные данные.

Входные данные 2 числа x_1 и x_2 .

Выходные данные ответ z .

2. Разрабатываем (ищем) методы решения задачи.

Методика проста, рассматривать нет необходимости.

3. Разрабатываем алгоритм решения задачи.

Алгоритм нарисуете самостоятельно.

4. Проектируем и настраиваем интерфейс программы: размещаем элементы для ввода исходных данных (обычно поля ввода **TextBox**), размещаем элементы для вывода ответа (обычно поля вывода **TextBox**), размещаем управляющие элементы (обычно кнопки **Button**), размещаем поясняющие подписи (обычно **Label**), настраиваем элементы интерфейса.

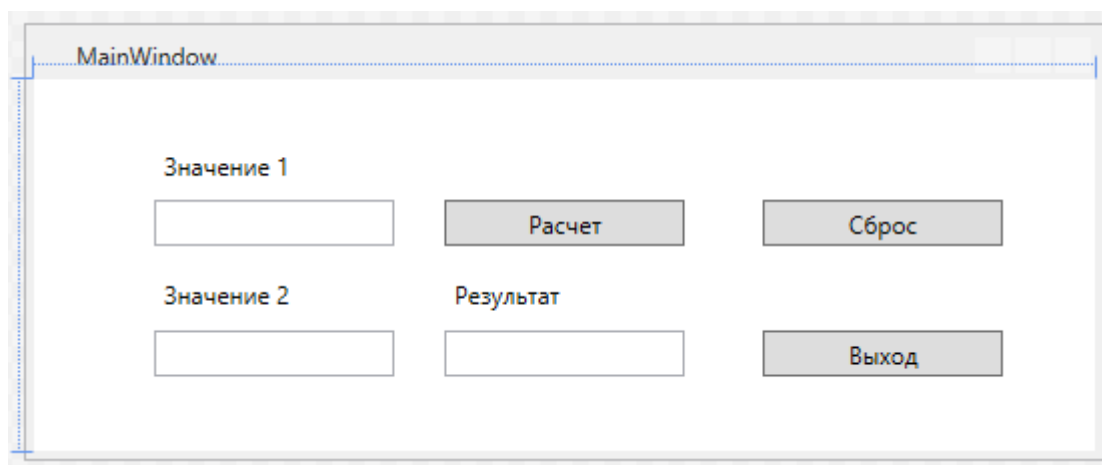


Рисунок. 9 - Пример окна программы

Таблица 1. Описание свойств настройки визуальных элементов программы

Свойство	Значение	Комментарий
Window – окно программы		
SizeMode	CanMinimize	Запретить изменять размер окна.
Icon	Значок	Выбрать значок для системного меню формы (левый верхний угол). Значки можно найти в папке C:\tools\Иконки и глифы\ico
Title	Сумма 2-х чисел	Заголовок окна
TextBox (tbZn1) – поле ввода 1 значения		
Имя (x:Name)	tbZn1	Задаем имя компонента для обращения к нему. Задать надо в окне свойств – поле Имя , или в разметке интерфейса атрибут x:Name .
MaxLength	9	Ограничиваем количество введенных знаков, т.к. тип int имеет ограниченный диапазон, чтобы не было ошибки при вычислении.
TextBox (tbZn2) – поле ввода 2 значения		
Имя (x:Name)	tbZn2	Задаем имя компонента для обращения к нему. Задать надо в окне свойств – поле Имя , или в разметке интерфейса атрибут x:Name .
MaxLength	9	
TextBox (tbRez) – поле вывода ответа		
Имя (x:Name)	tbRez	Задаем имя компонента для обращения к нему. Задать надо в окне свойств – поле Имя , или в разметке интерфейса атрибут x:Name .
isReadOnly	True	Блокируем поле, чтобы в него нельзя было вводить информацию, т.к. это поле ответа.
Button – кнопка расчет		
Content	Расчет	Название кнопки
isDefault	True	При нажатии клавиши Enter, нажметесь кнопка на форме, рассчитать.
Button – кнопка сброс		
Content	Сброс	Название кнопки
isCancel	True	При нажатии клавиши Esc, нажметесь кнопка на форме, сброс

Button – кнопка выход		
Content	Выход	Название кнопки
Label1 – поясняющие надписи		
Content	Значение 1, Значение 2, Результат	Поясняющие надписи

После создания и настройки интерфейса в окне разметки интерфейса в *XAML*, мы увидим примерно следующий код:

```
<Label Content="Значение 1" HorizontalAlignment="Left" Margin="60,30,0,0" VerticalAlignment="Top" Width="110"/>
<Label Content="Значение 2" HorizontalAlignment="Left" Margin="60,94,0,0" VerticalAlignment="Top" Width="110"/>
<TextBox x:Name="tbZn1" HorizontalAlignment="Left" Height="23" Margin="60,60,0,0" TextWrapping="Wrap"
    VerticalAlignment="Top" Width="120" MaxLength="9" TextChanged="tbZn1_TextChanged"/>
<TextBox x:Name="tbZn2" HorizontalAlignment="Left" Height="23" Margin="60,125,0,0" TextWrapping="Wrap"
    VerticalAlignment="Top" Width="120" MaxLength="9"/>
<TextBox x:Name="tbRez" HorizontalAlignment="Left" Height="23" Margin="205,125,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" MaxLength="9"/>
<Button Content="Рассчет" HorizontalAlignment="Left" Margin="205,60,0,0" VerticalAlignment="Top" Width="120" Height="23" Click="btnCalc_Click" IsDefault="True" />
<Button Content="Выход" HorizontalAlignment="Left" Margin="364,125,0,0" VerticalAlignment="Top" Width="120" Height="23" Click="btnExit_Click" />
<Button Content="Сброс" HorizontalAlignment="Left" Margin="364,60,0,0" VerticalAlignment="Top" Width="120" Height="23" Click="btnClear_Click" IsCancel="True" />
<Label Content="Результат" HorizontalAlignment="Left" Margin="205,94,0,0" VerticalAlignment="Top" Width="110"/>
```

Рисунок. 10. Код разметки интерфейса текущей задачи

5. Продумываем логику работу программы. Это события, на какие будет реагировать наша программы. Чаще всего это нажатие на кнопку **Рассчитать**. В более сложном случае взаимодействие событий и элементов программы.

Таблица 2. Настройка логики работы программы

Событие	Комментарий
Нажатие (Click) на кнопку Рассчитать	Расчёт по заданию
Нажатие (Click) на кнопку Сброс	Очищает все поля TextBox
Нажатие (Click) на кнопку Выход	Завершение работы программы

6. Разрабатываем обработчики событий, т.е. пишем программой код.

Так как событие **Click** для кнопки является основным, то создать или перейти в обработчик этого события можно двойным щелчком по элементу кнопка на форме программе. Для каждой кнопки создается свой обработчик события. Но такой способ создания события не совсем хорош т.к. название события получается не информативным **ButtonClick1**, **ButtonClick2** и т.д. и не соответствует стандартам оформления кода.

Для более корректного именования можно изначально задать имя кнопке **x:Name**, например, для кнопки расчет – **«btnCalc»** или **«btnCalculation»**. Также можно просто вручную изменить название обработчика события.

При создании события в файле кода программы появляется заготовка обработчика события, также в разметке интерфейса у кнопки появляется атрибут **Click="btnCalc_Click"**.

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

Рисунок. 11. Заготовка обработчика события

Также можно создать событие в окне *Свойства* на вкладке *События* двойным щелчком напротив нужного события см. рисунок 11, при этом создается событие со стандартным названием см. описание выше. Также можно написать название события и нажать *Enter*.

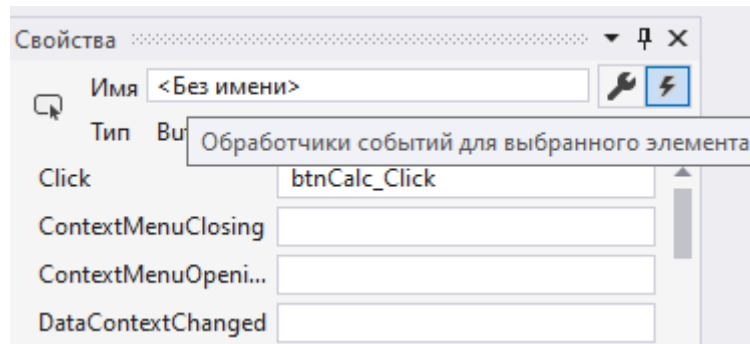


Рисунок. 11. Создание события

Код программы (обработчиков событий) будет выглядеть следующим образом:

```
private void btnCalc_Click(object sender, RoutedEventArgs e)
{
    int value1, value2, rez;
    //Получаем данные с формы
    value1 = Convert.ToInt32(tbZn1.Text);
    value2 = Convert.ToInt32(tbZn2.Text);
    rez = value1 + value2; //расчитываем
    //Выводим ответ на форму
    tbRez.Text = Convert.ToString(rez);
}
```

Ссылка: 1

```
private void btnExit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
```

Ссылка: 1

```
private void btnClear_Click(object sender, RoutedEventArgs e)
{
    tbZn1.Clear();
    tbZn2.Clear();
    tbRez.Clear();
    tbZn1.Focus();
}
```

Рисунок. 12. Код обработчиков событий

Тюнинг приложения для создания удобной и отказоустойчивой программы

Вы мы рассмотрели, как создать программу Windows. У нас получилась хорошая программа с удобным интерфейсом, можно радоваться, но на самом деле программа остается еще довольно кривой и неудобной в использовании. Рассмотрим далее, что тут не так.

1. Запустим программу и, не вводя исходные данные, нажмем кнопку рассчитать и увидим, что при выполнении программы возникла критичная ошибка и выполнение остановилось. Ошибка возникла в этой строчке кода:

```
value1 = Convert.ToInt32(tbZn1.Text);
```

т.е. программа не может пустой текст преобразовать в число. Что делать? Можно сделать так:

```
if (tbZn1.Text != "" && tbZn2.Text != "")
{
    value1 = Convert.ToInt32(tbZn1.Text);
    value2 = Convert.ToInt32(tbZn2.Text);
    rez = value1 + value2; //расчитываем
                        //Выводим ответ на форму
    tbRez.Text = Convert.ToString(rez);
}
else MessageBox.Show("Ведите правильные значения");
```

2. Запустим программу и введем вместо цифр буквы, нажмем кнопку рассчитать и увидим, что при выполнении программы возникла критичная ошибка и выполнение остановилось. Ошибка возникла в этой строчке кода:

```
value1 = Convert.ToInt32(tbZn1.Text);
```

т.е. программа не может текст преобразовать в число. Что делать? Можно использовать следующую функцию для ввода значения:

```
bool TryParse(string s, out int result)
```

Параметры

s - строка, содержащая преобразуемое число.

result - целочисленное значение, эквивалентное числу, содержащемуся в параметре *s*, если преобразование выполнено успешно, или нуль, если оно завершилось неудачей.

Возвращаемое значение

Значение *true*, если параметр *s* успешно преобразован; в противном случае — значение *false*.

Код программы будет выглядеть следующим образом:

```
int value1, value2, rez;
bool f1, f2;
//Получаем данные с формы
f1 = Int32.TryParse(tbZn1.Text, out value1);
f2 = Int32.TryParse(tbZn2.Text, out value2);
if (f1 == true && f2 == true)
{
    rez = value1 + value2; //расчитываем
    //Выводим ответ на форму
    tbRez.Text = Convert.ToString(rez);
}
else MessageBox.Show("Ведите правильные значения");
```

3. Введем исходные значения 2 и 2, нажмем кнопку *Рассчитать* и увидим ответ 4, т.е. все правильно. Изменяем исходные значения на 3 и 3 и не жмем кнопку Рассчитать и видим, что $3 + 3 = 4$. Хорошо бы при изменении исходных данных очищать поле результата.

Таблица 3. Логика работы очистки поля результата

Событие	Комментарий
Изменение содержимого поля значение 1 (TextChanged)	Очищаем поле результата
Изменение содержимого поля значение 2 (TextChanged)	Очищаем поле результата

Код программы будет выглядеть следующим образом:

```
//При изменении значений, очищать результат
Ссылка: 1
private void tbZn1_TextChanged(object sender, TextChangedEventArgs e)
{
    tbRez.Clear();
}
```

4. Нажмем кнопку Сброс, исходные значения очищаются вроде бы все хорошо, но для ввода новых значений нужно лишний раз мышкой щелкнуть на поле исходных данных. Можно после очистки полей передать фокус в первое поле результата. Изменяем обработчик кнопки Сброс следующим образом.

```
private void btnClear_Click(object sender, RoutedEventArgs e)
{
    tbZn1.Clear();
    tbZn2.Clear();
    tbRez.Clear();
    tbZn1.Focus();
}
```


Практическая работа №17

Создание проекта простого оконного приложения

Начальные условия

1. Размер шрифта 12 пт.
2. Для разделения заданий в одном окне использовать панель (**GroupBox**), не забудьте в панель вставить контейнер **Grid** иначе собирать элементы в **GroupBox** будет проблематично.
3. Обеспечить защиту полей ввода, от ввода некорректных значений.
4. Обеспечить неизменяемость границ основного окна.
5. Добавить иконку в заголовок программы и исполняемый файл.
6. Заблокировать ввод данных в поле вывода информации.
7. При изменении исходных значений очищать поля вывода.
8. Предусмотреть в программе две кнопки «**Выход**» и «**О программе**», где вывести ФИО разработчика, номер работы и формулировку задания.
9. Программу оформить комментариями.

Варианты заданий (Базовый уровень)

1. Реализовать расчет задачи:
 - Дана сторона квадрата a . Найти его площадь и периметр.
 - Дано трехзначное число. Вывести вначале его последнюю цифру (единицы), а затем — его среднюю цифру (десятки).
2. Реализовать расчет задачи:
 - Даны стороны прямоугольника a и b . Найти его площадь и периметр.
 - Дано трехзначное число. Вывести число, полученное при прочтении исходного числа справа налево.
3. Реализовать расчет задачи:
 - Дан диаметр окружности D . Найти ее длину $L = \pi \cdot D$. В качестве значения π использовать 3.14.
 - Дано трехзначное число. В нем зачеркнули первую справа цифру и приписали ее слева. Вывести полученное число.
4. Реализовать расчет задачи:
 - Дана длина ребра куба A . Найти объем куба V и площадь его поверхности S .
 - Дана масса M в килограммах. Используя операцию деления целых чисел, найти количество полных тонн и килограмм (1 тонна = 1000 кг).
5. Реализовать расчет задачи:
 - Даны длины ребер a , b , c прямоугольного параллелепипеда. Найти его объем $V = a \cdot b \cdot c$ и площадь поверхности $S = 2 \cdot (a \cdot b + b \cdot c + a \cdot c)$.
 - Дано двузначное число. Найти сумму и произведение его цифр.
6. Реализовать расчет задачи:
 - Даны два неотрицательных числа a и b . Найти их среднее геометрическое, то есть квадратный корень из их произведения.
 - Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа (например, 123 перейдет в 213).
7. Реализовать расчет задачи:
 - Даны катеты прямоугольного треугольника a и b . Найти его гипотенузу c и периметр P .
 - Дано целое число, большее 999. Используя одну операцию деления нацело и одну операцию взятия остатка от деления, найти цифру, соответствующую разряду сотен в записи этого числа.

8. Реализовать расчет задачи:
 - Даны два круга с общим центром и радиусами R_1 и R_2 ($R_1 > R_2$). Найти площади этих кругов S_1 и S_2 , а также площадь S_3 кольца, внешний радиус которого равен R_1 , внутренний радиус равен R_2 : $S_1 = \pi \cdot (R_1)^2$, $S_2 = \pi \cdot (R_2)^2$, $S_3 = S_1 - S_2$. В качестве значения π использовать 3.14.
 - Дан номер некоторого года (целое положительное число). Определить соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год.
9. Реализовать расчет задачи:
 - Дана площадь S круга. Найти его диаметр D и длину L окружности, ограничивающей этот круг, учитывая, что $L = \pi \cdot D$, $S = \pi \cdot D^2 / 4$. В качестве значения π использовать 3.14.
 - С начала суток прошло N секунд (N — целое). Найти количество полных часов, прошедших сначала суток.
10. Реализовать расчет задачи:
 - Даны три точки A , B , C на числовой оси. Найти длины отрезков AC и BC и их сумму.
 - С начала суток прошло N секунд (N — целое). Найти количество полных минут, прошедших сначала последнего часа.
11. Реализовать расчет задачи:
 - Даны координаты двух противоположных вершин прямоугольника: (x_1, y_1) , (x_2, y_2) . Стороны прямоугольника параллельны осям координат. Найти периметр и площадь данного прямоугольника.
 - Дан размер файла в байтах. Используя операцию деления нацело, найти количество полных килобайтов, которые занимает данный файл (1 килобайт = 1024 байта).
12. Реализовать расчет задачи:
 - Найти расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2) на плоскости. Расстояние вычисляется по формуле $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
 - Дано трехзначное число. Используя одну операцию деления нацело, вывести первую цифру данного числа (сотни).
13. Реализовать расчет задачи:
 - Даны координаты трех вершин треугольника: (x_1, y_1) , (x_2, y_2) , (x_3, y_3) . Найти его периметр и площадь, используя формулу для расстояния между двумя точками на плоскости (см. задание 12). Для нахождения площади треугольника со сторонами a , b , c использовать формулу Герона: $S = \sqrt{p(p-a)(p-b)(p-c)}$ где $p = (a + b + c) / 2$ — полупериметр.
 - Дано трехзначное число. Найти сумму и произведение его цифр.
14. Реализовать расчет задачи:
 - Даны три точки A , B , C на числовой оси. Точка C расположена между точками A и B . Найти произведение длин отрезков AC и BC .
 - Дано трехзначное число. Найти сумму и произведение его цифр.
15. Реализовать расчет задачи:
 - Найти значение функции $y = 3 \cdot x^6 - 6 \cdot x^2 - 7$ при данном значении x .
 - Даны целые положительные числа A и B ($A > B$). На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений). Используя операцию деления нацело, найти количество отрезков B , размещенных на отрезке A .
16. Реализовать расчет задачи:
 - Найти значение функции $y = 4 \cdot (x-3)^6 - 7 \cdot (x-3)^3 + 2$ при данном значении x .

- С начала суток прошло N секунд (N — целое). Найти количество секунд, прошедших с начала последней минуты.

17. Реализовать расчет задачи:

- Дано значение температуры T в градусах Цельсия. Определить значение этой же температуры в градусах Фаренгейта. Температура по Цельсию T_C и температура по Фаренгейту T_F связаны следующим соотношением: $T_C = (T_F - 32) \cdot 5/9$.
- Дано двузначное число. Вывести число, полученное при перестановке цифр исходного числа.

18. Реализовать расчет задачи:

- Известно, что X кг шоколадных конфет стоит A рублей, а Y кг ирисок стоит B рублей. Определить, сколько стоит 1 кг шоколадных конфет, 1 кг ирисок, а также во сколько раз шоколадные конфеты дороже ирисок.
- Дано расстояние L в сантиметрах. Используя операцию деления целых чисел, найти количество полных метров и сантиметров (1 метр = 100 см).