

Основы программирования

Введение в разработку Windows-приложений

Введение в Windows Forms

- Консольное приложение.
- Приложение Windows Forms.
- Веб-приложение.
- ...

Событийное программирование

В основу Windows положен принцип событийного управления.





Система, и приложения после запуска ожидают действий пользователя и реагируют на них заранее заданным образом.

Любое действие пользователя (нажатие клавиши на клавиатуре, щелчок кнопкой мыши, перемещение мыши) называется событием.

Создание проекта

Создание проекта

Последние шаблоны проектов

-  Приложение Windows Forms (.NET Framework) C#
-  Веб-приложение ASP.NET Core C#
-  Консольное приложение (.NET Core) C#
-  Консольное приложение (.NET Core) Visual Basic

Поиск шаблонов (ALT+"B") 🔍

Все языки ▾

Все платформы ▾

Все типы проектов ▾

C++

Windows

Консоль



Приложение Windows Forms (.NET Framework)

Проект для создания приложения с пользовательским интерфейсом Windows Forms (WinForms)

C#

Windows

Рабочий стол



Windows Forms App (.NET)

Проект для создания приложения с пользовательским интерфейсом Windows Forms (WinForms)

Visual Basic

Windows

Рабочий стол



Проект CMake

Создавайте современные кроссплатформенные приложения C++, не зависящие от файлов SLN или VCXPROJ.

C++

Windows

Linux

Консоль



Мастер классических приложений Windows

Назад

Далее

Создание проекта

Настроить новый проект


Приложение Windows Forms (.NET Framework) C# Windows Рабочий стол

Имя проекта

WindowsFormsApp2

Расположение

C:\Users\Svetlana\source\repos

Имя решения 

WindowsFormsApp2

☐ Поместить решение и проект в одном каталоге

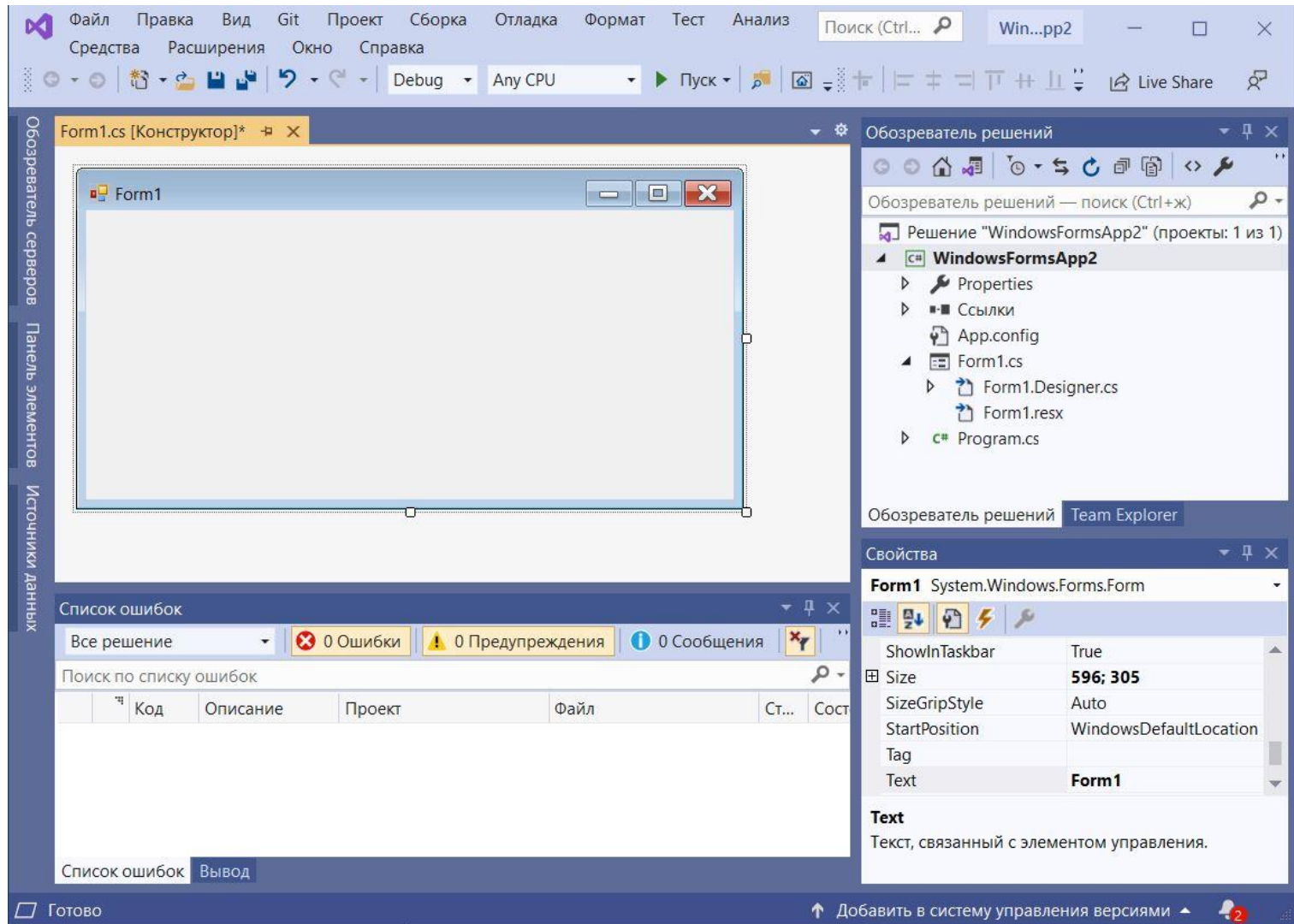
Платформа

.NET Framework 4.7.2

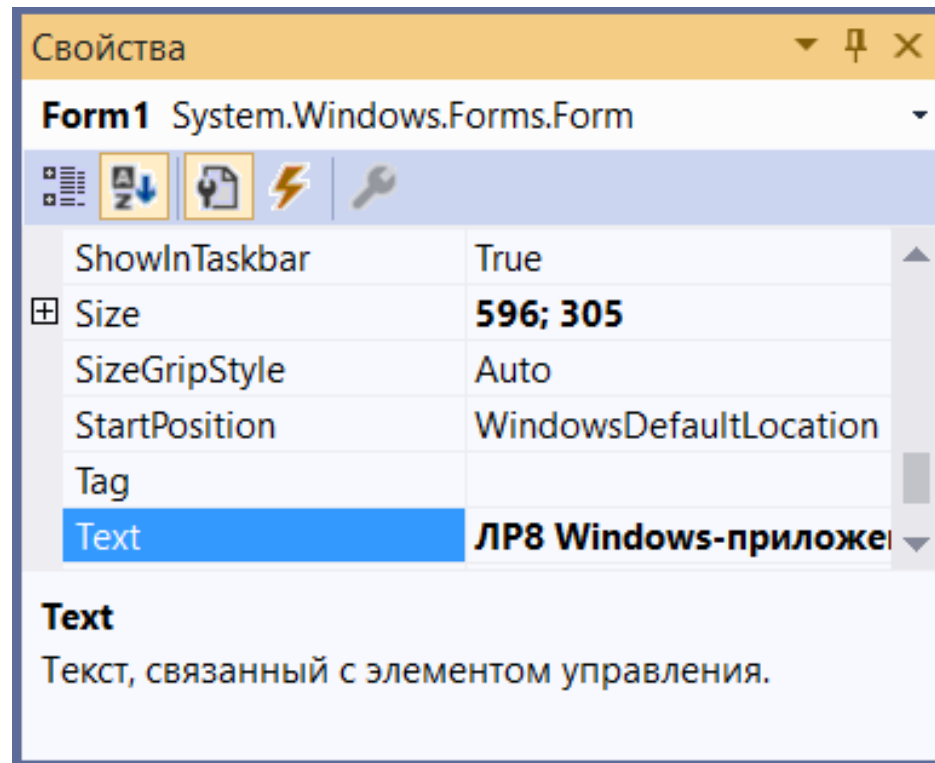
Назад

Создать

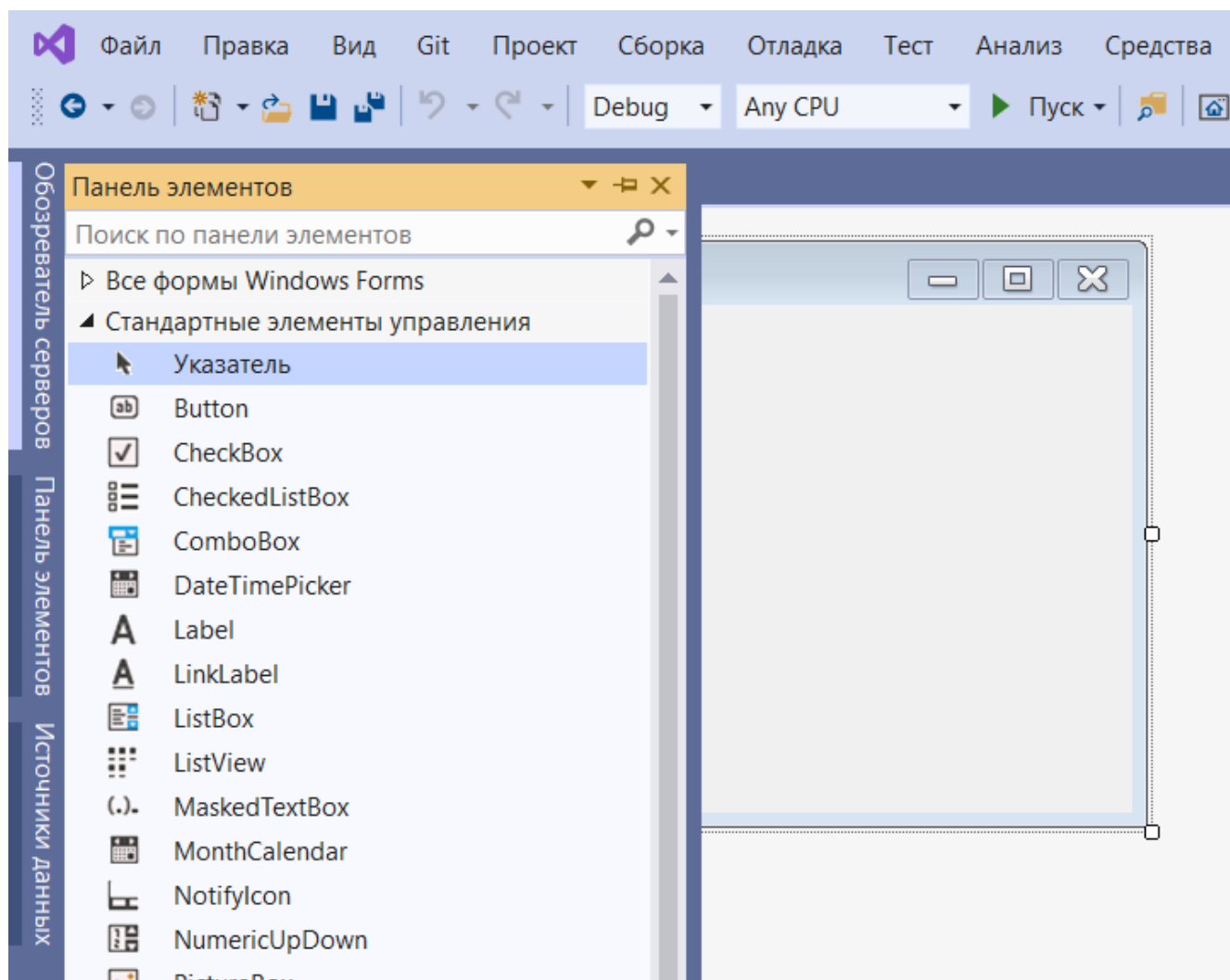
Стандартный проект открыт



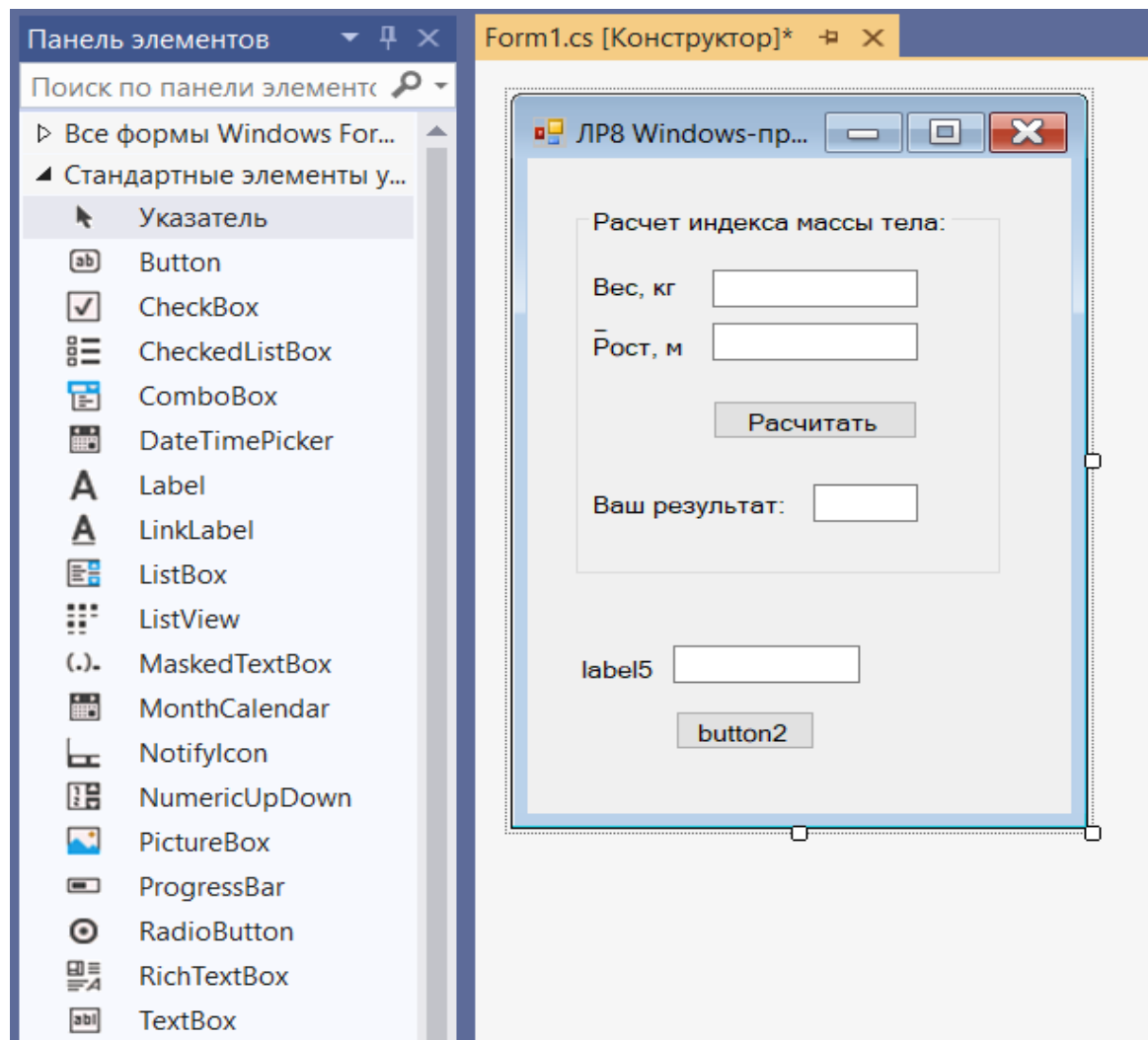
Изменение названия формы



Панель элементов



Выбор и размещение элемента на форме

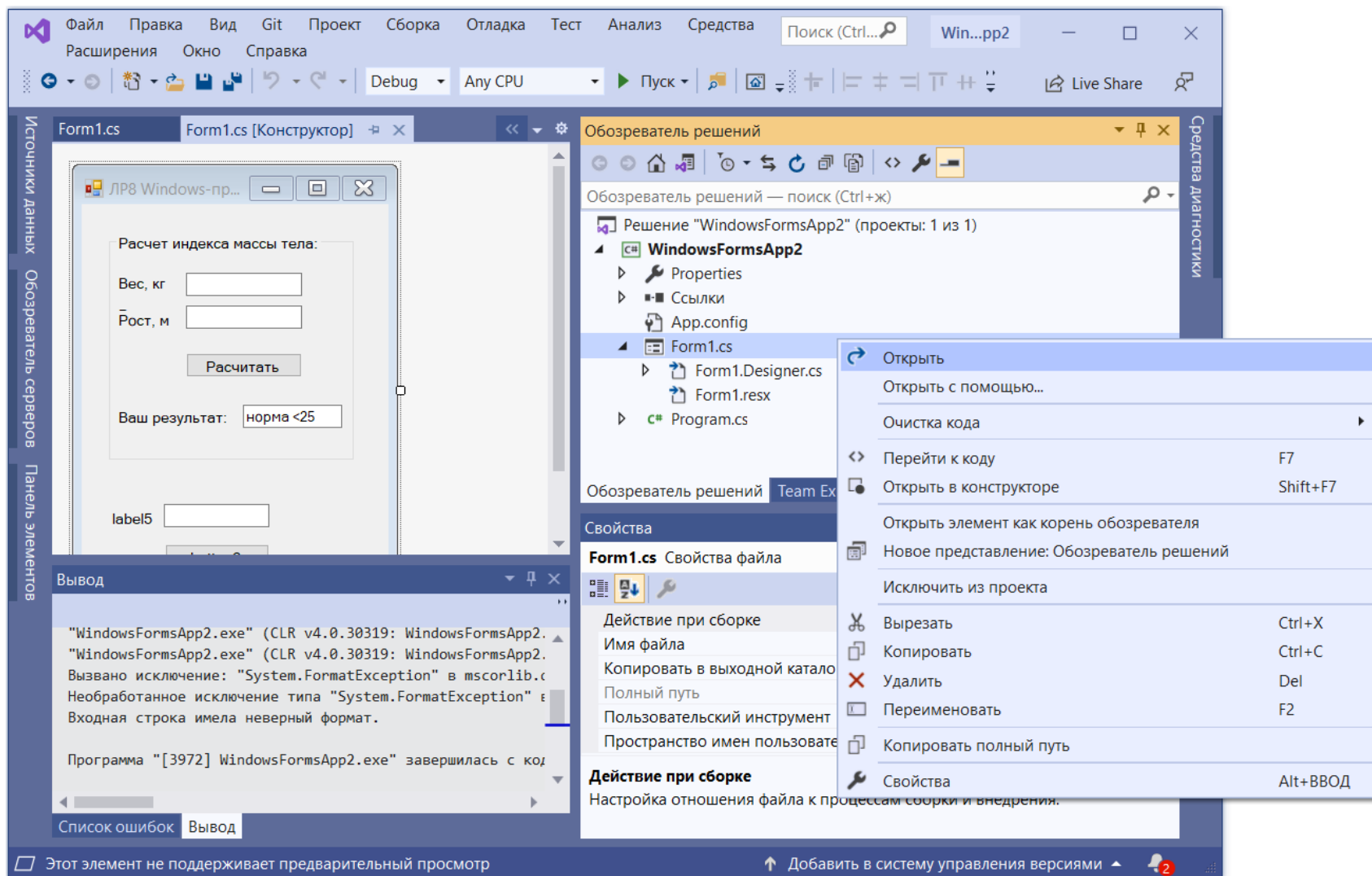


Просмотр кода

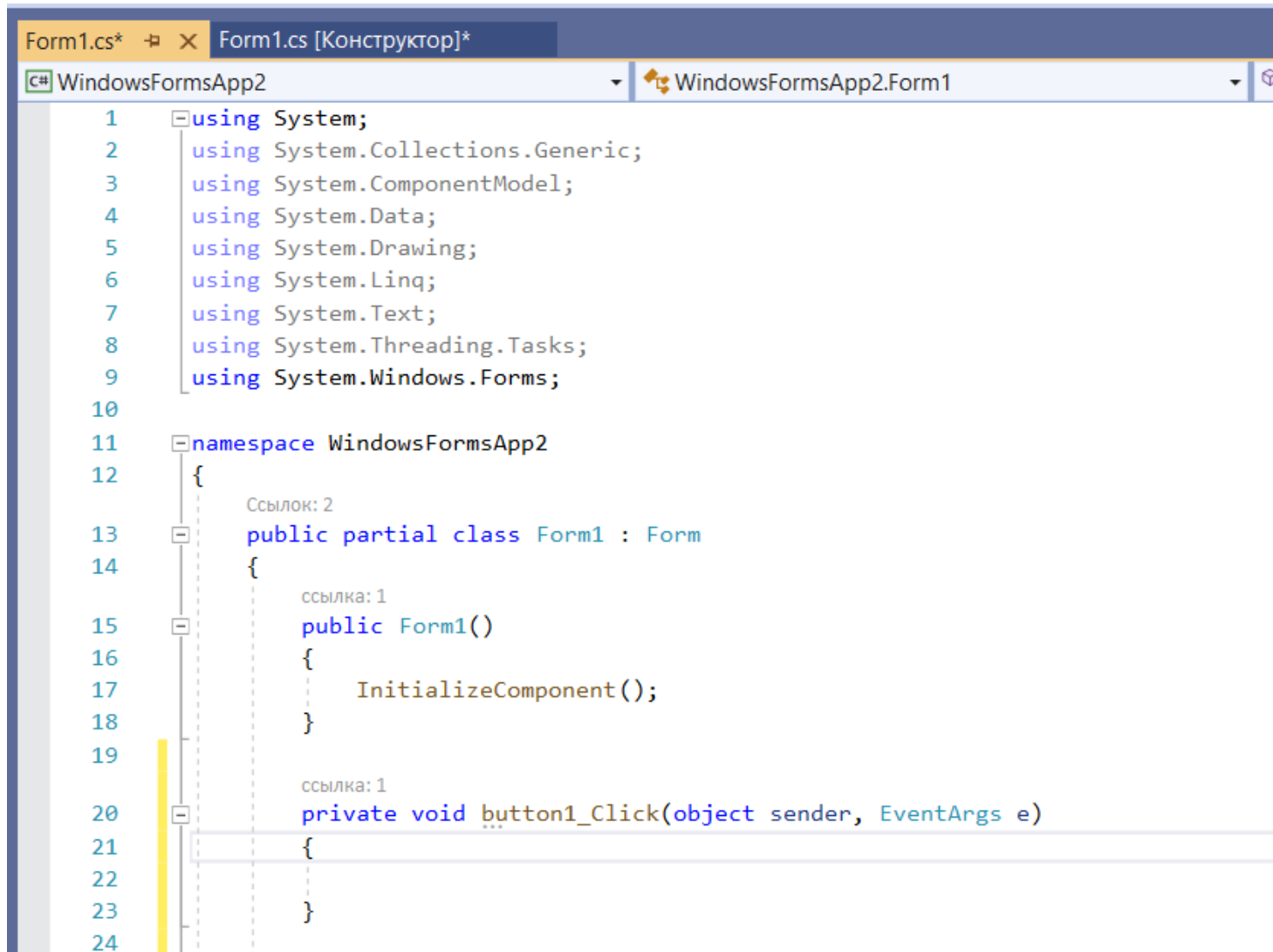
Открыть файл кода можно так:

- щелкнуть форму правой кнопкой мыши и в появившемся меню выбрать «Перейти к коду» (View Code);
- F7:
- В «Обозревателе решений» щелкнуть форму правой кнопкой мыши (на `Form1.cs`) и в появившемся меню выбрать «Перейти к коду»;

Окно «Обозреватель решений»



Просмотр кода



```
Form1.cs*  Form1.cs [Конструктор]*
C# WindowsFormsApp2  WindowsFormsApp2.Form1
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace WindowsFormsApp2
12 {
13     Ссылка: 2
14     public partial class Form1 : Form
15     {
16         ссылка: 1
17         public Form1()
18         {
19             InitializeComponent();
20         }
21         ссылка: 1
22         private void button1_Click(object sender, EventArgs e)
23         {
24         }
```

Код примера

```
11 namespace WindowsFormsApp2
12 {
13     Ссылка: 2
14     public partial class Form1 : Form
15     {
16         ссылка: 1
17         public Form1()
18         {
19             InitializeComponent();
20
21         ссылка: 1
22         private void button1_Click(object sender, EventArgs e)
23         {
24             double m = Convert.ToDouble(textBox1.Text);
25             double h = Convert.ToDouble(textBox2.Text);
26             double i = m * 10000 / Math.Pow(h, 2);
27             //textBox3.Text = Convert.ToString(i);
28             textBox3.Text = String.Format("{0,5:#0.00}", i);
29         }
30     }
31 }
```

Запуск приложения

- Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio.
- После запуска приложения Visual Studio компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release.

Как это работает

The image displays a Visual Studio IDE with two windows. The left window shows the source code for `Form1.cs`, which is a partial class for a Windows Form. The code includes a constructor and two event handlers. The right window shows a preview of the application running, titled "ЛР8 Windows-приложение".

Source Code (Form1.cs):

```
12 {  
13     Ссылка: 2  
14     public partial class Form1 : Form  
15     {  
16         Ссылка: 1  
17         public Form1()  
18         {  
19             InitializeComponent();  
20         }  
21         Ссылка: 1  
22         private void button1_Click(object sender, EventArgs e)  
23         {  
24             double m = Convert.ToDouble(textBox1.Text);  
25             double h = Convert.ToDouble(textBox2.Text);  
26             double i = m * 10000 / Math.Pow(h, 2);  
27             //textBox3.Text = Convert.ToString(i);  
28             textBox3.Text = String.Format("{0,5:#0.00}", i);  
29         }  
30         Ссылка: 1  
31         private void groupBox1_Enter(object sender, EventArgs e)  
32         {  
33         }  
34     }  
}
```

Application Preview (ЛР8 Windows-приложение):

Расчет индекса массы тела:

Вес, кг:

Рост, см:

Ваш результат:

label5:

Код примера (2)

```
11 namespace WindowsFormsApp2
12 {
13     Ссылка: 2
14     public partial class Form1 : Form
15     {
16         Ссылка: 1
17         public Form1()
18         {
19             InitializeComponent();
20             //this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(255)))), ((int)(((byte)(192)))));
21         }
22         Ссылка: 1
23         private void button1_Click(object sender, EventArgs e)
24         {
25             double m = Convert.ToDouble(textBox1.Text);
26             double h = Convert.ToDouble(textBox2.Text);
27             double i = m * 10000 / Math.Pow(h,2);
28             //textBox3.Text = Convert.ToString(i);
29             //this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(192)))), ((int)(((byte)(255)))));
30             textBox3.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(255)))), ((int)(((byte)(192)))));
31             textBox3.Text = String.Format("{0,5:#0.00}", i);
32         }
33     }
34 }
35 Ссылка: 1
```


Пример

Калькулятор
"Индекс массы тела (ИМТ)"

Ваш рост, в см.

170

Ваш вес, в кг.

50

ИМТ: 17.3

Недостаточная масса тела.
Соответствует ИМТ 16 – 18,5

Заключение:

До нормы, Вам не хватает 3.5 кг.
Ваш нормальный вес должен
находиться в пределах 53.5 – 72.2 кг.

Некоторые полезные элементы управления

- Label (Надпись).
- Button (Кнопка).
- ListBox (Список).
- CheckBox (Флажок).
- RadioButton (Переключатель).
- MessageBox (Окно сообщений).
- Menu (Меню).
- Toolbar (Панель инструментов).
- DataGrid (Сетка данных).
- ...

Свойства элементов управления

- `Name` – определяет имя элемента управления;
- `BackColor` – определяет фоновый цвет элемента;
- `Enabled` – определяет, будет ли доступен элемент для использования. Если это свойство имеет значение `False`, то элемент блокируется
- `Visible` – определяет, будет ли доступен элемент виден;
- `Size` – определяет размер элемента;
- `Width` и `Height` – определяют ширину и высоту элемента;
- `TabIndex` – определяет порядок обхода элемента по нажатию на клавишу `Tab`;
- ...

Шаблон MVC

Model-View-Controller (MVC,
«Модель-Представление-Контроллер»,
«Модель-Вид-Контроллер»)

Это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента:

- модель,
- представление
- контроллер

таким образом, что модификация каждого компонента может осуществляться независимо.

Шаблон MVC

Обработка данных и логика приложения

➤ **Model**

Представление данных пользователю в
любом поддерживаемом формате

➤ **View**

Обработка запросов пользователя и
вызов соответствующих ресурсов

➤ **Controller**

Шаблон MVC. Представление/Вид

- **Представление (View)** обеспечивает различные способы представления данных модели, в зависимости от состояния модели.
- Это может быть шаблон, который заполняется данными.
- Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Шаблон MVC. Модель

- **Модель (Model)** - предоставляет данные и правила, которые используются для работы с данными, реагирует на команды контроллера, изменяя своё состояние.
- Модель передает контроллеру данные, которые запросил пользователь. Модель данных будет одинаковой, вне зависимости от того, как мы хотим представлять их пользователю.

Шаблон MVC. Контроллер

- **Контроллер (Controller)** обрабатывает действия пользователя (когда пользователь нажимает на элементы интерфейса) оповещая модель о необходимости изменений.
- Его основная функция — вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

Назначение шаблона MVC

- Основная цель этой концепции состоит в отделении бизнес-логики (*модели*) от её визуализации (*представления, вида*).
- За счёт такого разделения повышается возможность повторного использования кода.
- Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

Решаются следующие задачи:

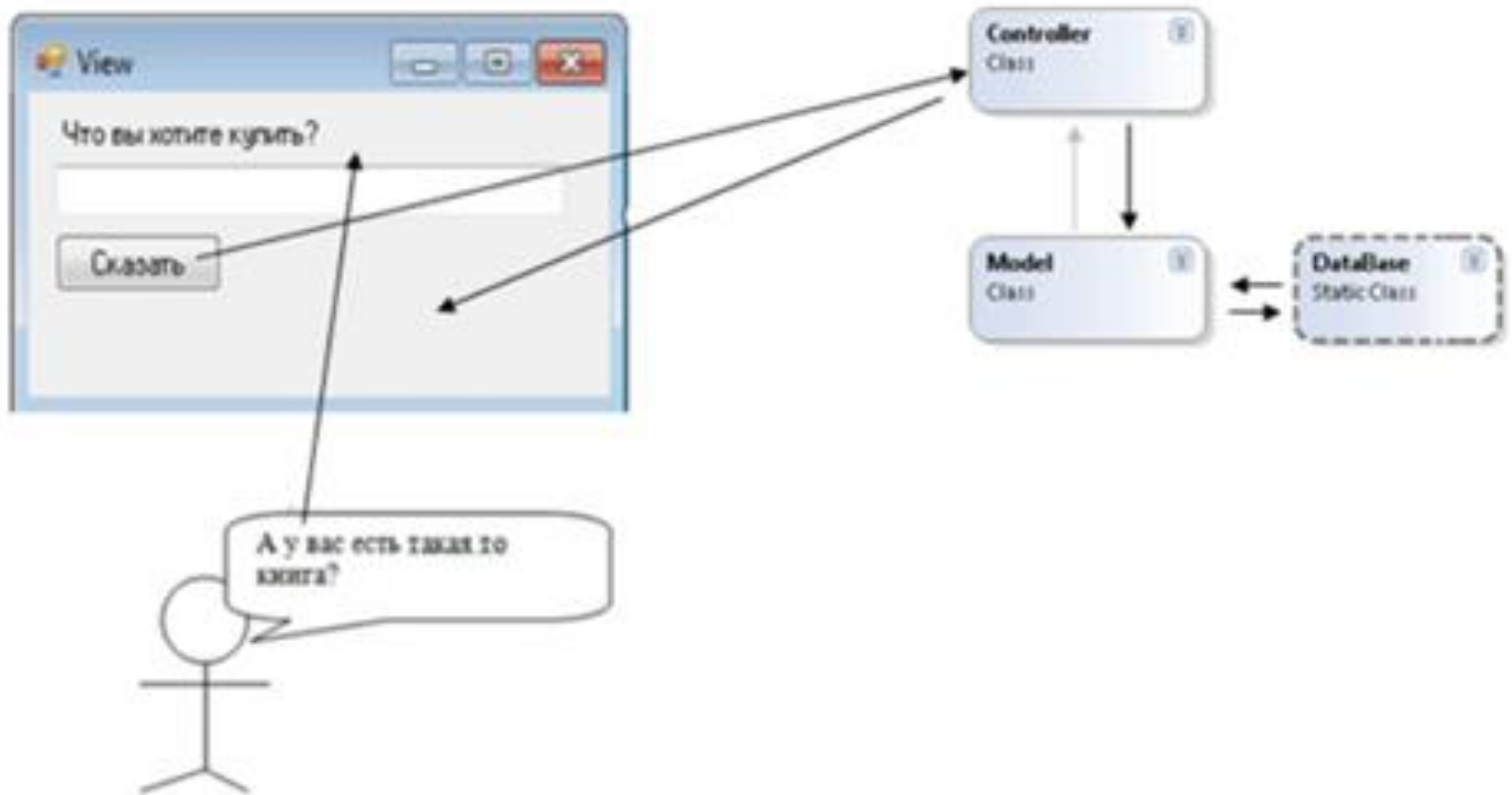
- к одной *модели* можно присоединить несколько *видов*, (реализация *модели* не затрагивается).
Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
- не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой *контроллер*;
- возможность добиться того, что программистам, занимающиеся разработкой бизнес-логики (*модели*), не нужны подробности о том, какое *представление* будет использоваться, так как часто разработчики специализируется либо на разработке пользовательского интерфейса, либо бизнес-логики.

Пример

«Поиск книги в книжном магазине»

1. Представлением будет форма, на которой есть поле, куда пользователь будет вводить вопрос.
2. Представление этот вопрос посылает контроллеру.
3. Контроллер посылает запрос модели.
4. После получения результата, контроллер дает ответ представлению.
5. Уже в самом представлении, конечный пользователь увидит ответ!

Как работает MVC



Рекомендованные источники

1. Руководство по программированию в Windows Forms
<https://metanit.com/sharp/windowsforms/>
2. Создание графических программ на C# и .NET
<https://metanit.com/sharp/forms.php>
3. Создание проекта приложения Windows Forms
<https://docs.microsoft.com/ru-ru/visualstudio/ide/step-1-create-a-windows-forms-application-project?view=vs-2019>
4. Основы паттернов проектирования
<https://metanit.com/sharp/patterns/1.1.php>
5. C# - Паттерн MVC (Model View Controller)
<https://webformymself.com/19-patterny-proektirovaniya-mvc-teoriya/>