Пошаговое руководство. Создание простого приложения с помощью Visual C# или Visual Basic

Выполняя данное пошаговое руководство, вы ознакомитесь со многими инструментами, диалоговыми окнами и конструкторами, которые можно использовать для создания приложений с помощью Visual Studio. Вам предстоит создать простое приложение типа “Hello, World”, разработать пользовательский интерфейс, добавить код и отладить ошибки, одновременно приобретая навыки работы в интегрированной среде разработки (IDE).

В этом разделе содержатся следующие подразделы.

[Настройка интегрированной среды разработки (IDE)](https://msdn.microsoft.com/ru-ru/library/jj153219.aspx#BKMK_ConfigureIDE)

[Создание простого приложения](https://msdn.microsoft.com/ru-ru/library/jj153219.aspx#BKMK_CreateApp)

[Отладка и тестирование приложения](https://msdn.microsoft.com/ru-ru/library/jj153219.aspx#BKMK_DebugTest)

|  |
| --- |
|  |
|  |

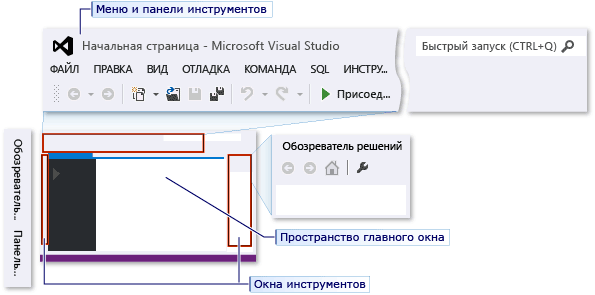
**Настройка интегрированной среды разработки (IDE)**

При первом запуске Visual Studio будет предложено войти в это приложение с использованием учетной записи Майкрософт: [Sign into Visual Studio](http://blogs.msdn.com/b/visualstudio/archive/2013/06/28/welcome-sign-in-to-visual-studio.aspx) (Вход в Visual Studio). Необязательно входить сразу, можно сделать это позже.

После запуска Visual Studio далее необходимо выбрать сочетание предопределенных параметров интегрированной среды разработки. Каждое сочетание параметров было разработано так, чтобы сделать разработку приложений более простой.

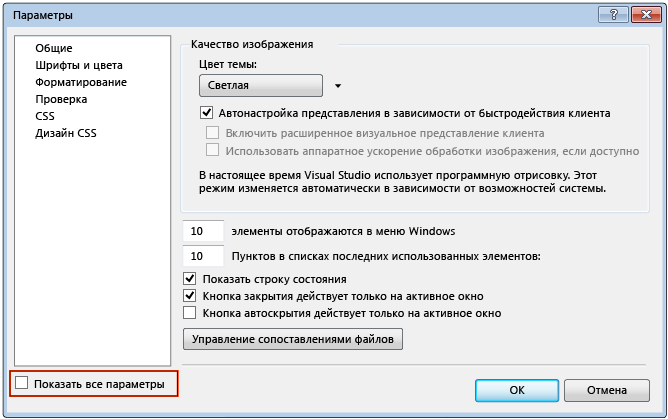
Это пошаговое руководство предполагает, что действуют **Обычные параметры разработки**, что соответствует минимальному объему настройки интегрированной среды разработки. Если вы уже выбрали C# или Visual Basic (подходит любой вариант), нет необходимости изменять свои настройки. Если вы хотите изменить настройки, можно воспользоваться **Мастером импорта и экспорта параметров**. Дополнительные сведения см. в разделе [Настройка параметров разработки в Visual Studio](https://msdn.microsoft.com/ru-ru/library/zbhkx167.aspx).

После открытия Visual Studio вы увидите окна инструментов, меню и панели инструментов, а также основную область окна. Окна инструментов закреплены в левой и правой частях окна приложения, а панель **Быстрый запуск**, строка меню и стандартная панель инструментов закреплены в верхней его части. В центре окна приложения находится **Начальная страница**. При загрузке решения или проекта редакторы и конструкторы отображаются в области **Начальной страницы**. При разработке приложения чаще всего используется именно эта область.



**Рис.** Интегрированная среда разработки Visual Studio

С помощью диалогового окна **Параметры** можно дополнительно настроить Visual Studio, например изменить в редакторе начертание и размер шрифта текста или изменить цветовую тему интегрированной среды разработки. В зависимости от примененного сочетания параметров некоторые элементы в этом диалоговом окне могут не отображаться автоматически. Чтобы отображались все возможные параметры, установите флажок **Показать все параметры**.

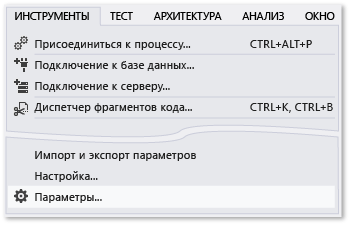


Диалоговое окно "Параметры"

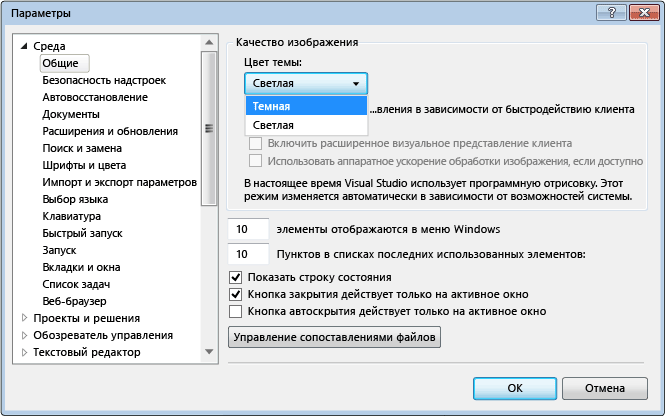
В этом примере вам предстоит изменить цветовую тему интегрированной среды разработки со светлой на темную. При желании можно сразу перейти к созданию проекта.

**Изменение цветовой темы интегрированной среды разработки**

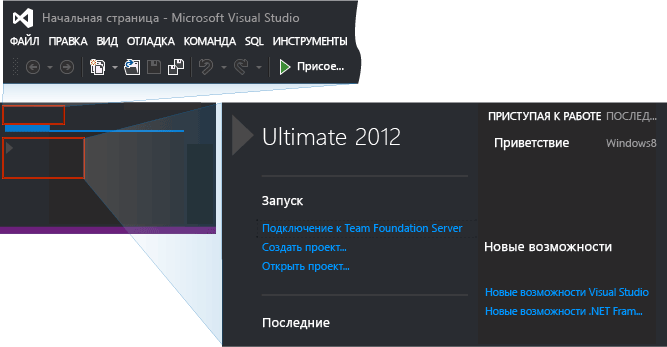
1. Откройте диалоговое окно **Параметры**, выбрав в меню **Сервис** вверху пункт **Параметры...**.



1. Измените **Цветовую тему** на **Темную**, а затем щелкните **ОК**.



Цвета в Visual Studio должны соответствовать следующему рисунку:



На изображениях в остальной части этого пошагового руководства используется светлая тема.

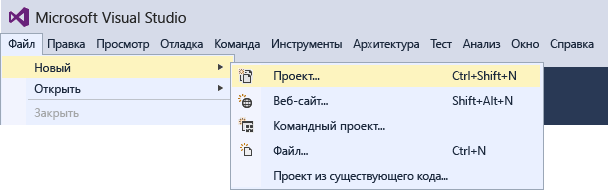
Создание простого приложения

**Создание проекта**

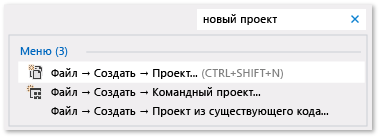
При создании приложения в Visual Studio необходимо сначала создать проект и решение. Этот пример демонстрирует создание решения Windows Presentation Foundation.

**Создание проекта WPF**

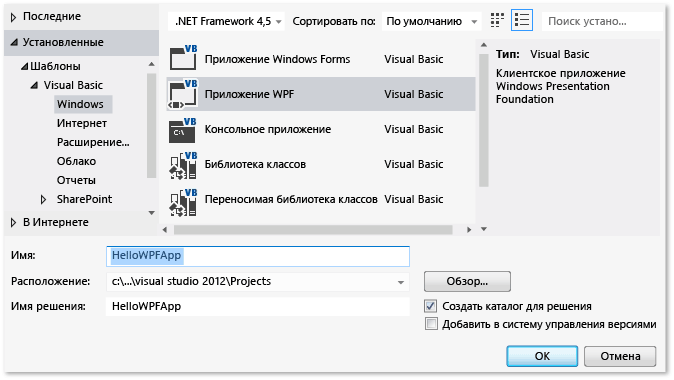
1. Создайте новый проект. В меню последовательно выберите пункты **Файл**, **Создать**, **Проект...**.



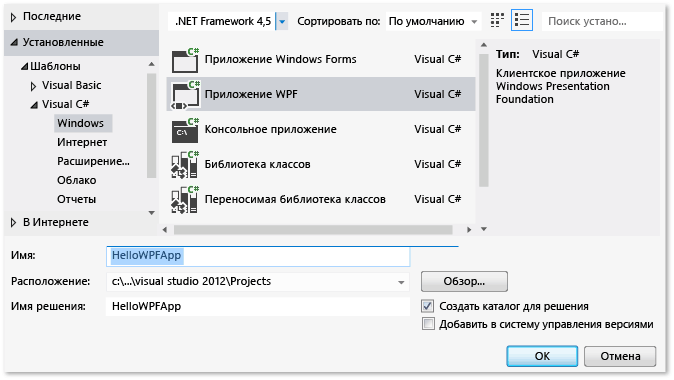
Кроме того, можно ввести **Создать проект** в окне **Быстрый запуск**, чтобы выполнить то же самое действие.



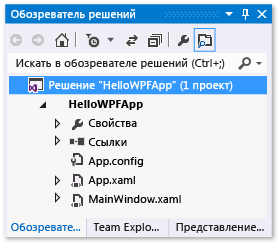
1. Выберите шаблон приложения WPF Visual Basic или Visual C# , воспользовавшись, например, путем по меню в левой области **Установлены**, **Шаблоны**, **Visual C#**, **Windows**, а затем выбрав «Приложение WPF» в средней области. В нижней части диалогового окна нового проекта назовите проект HelloWPFApp.



ИЛИ



Visual Studio создает решение и проект HelloWPFApp, и **Обозреватель решений** показывает различные файлы. Конструктор WPF отображает представление конструирования и представление XAML файла MainWindow.xaml в одном разделенном представлении. Сдвигая разделитель, можно делать любое из представлений больше или меньше. Можно выбрать для просмотра только визуальное представление или только представление XAML. В **Обозревателе решений** отображаются следующие элементы.

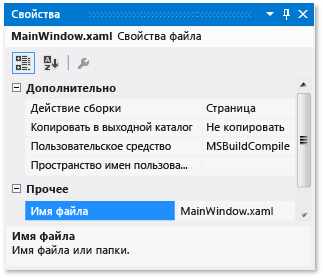


**Рис**. Элементы проекта

После создания проекта его можно настраивать. С помощью окна **Свойства** (в меню **Вид**) можно отображать и изменять параметры элементов проекта, элементов управления и других элементов в приложении. С помощью свойств проекта и страниц свойств можно отображать и изменять параметры проектов и решений.

**Изменение имени MainWindow.xaml**

1. В следующей процедуре вы дадите MainWindow более конкретное имя. В **Обозревателе решений** выберите файл MainWindow.xaml. Должно отображаться окно **Свойства**, но если его нет, выберите в меню **Вид** пункт **Окно свойств**. Измените значение свойства **Имя файла** на Greetings.xaml.



**Обозреватель решений** показывает, что файл теперь называется Greetings.xaml, и если развернуть узел MainWindow.xaml (переместив курсор на узел и нажав клавишу стрелки вправо), то видно, что файл MainWindow.xaml.vb (или MainWindow.xaml.cs) теперь называется Greetings.xaml.vb (или Greetings.xaml.cs). Этот файл с текстом программы вложен в узел файла .xaml, что означает их тесную связь.

|  |
| --- |
| **System_CAPS_ICON_warning.jpg Предупреждение** |
| Это изменение вызовет ошибку; сведения по ее отладке и исправлению будут предоставлены позднее. |

1. В **Обозревателе решений** откройте файл Greetings.xaml в представлении конструктора (путем нажатия клавиши ВВОД при выбранном узле) и выберите заголовок окна с помощью мыши.
2. В окне **Свойства** измените значение свойства **Заголовок** на Greetings.

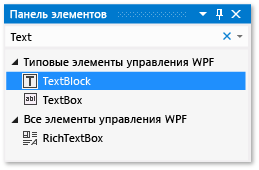
Заголовок окна для MainWindow.xaml теперь содержит текст Greetings.

**Конструирование пользовательского интерфейса (ИП)**

В приложение будет добавлено три типа элементов управления: элемент управления TextBlock, два элемента управления RadioButton, и элемент управления Button.

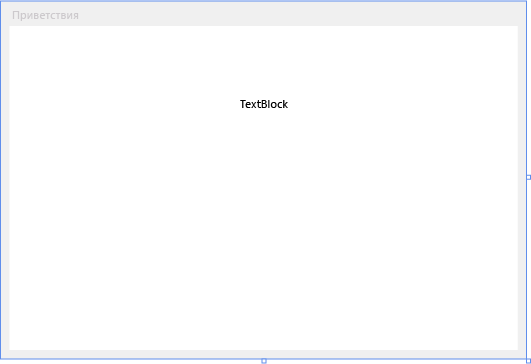
Добавление элемента управления **TextBlock**

1. Откройте окно **Панель элементов**, выбрав в меню **Вид** пункт **Панель элементов**.
2. На **Панели элементов** найдите элемент управления TextBlock.



1. Добавьте элемент управления TextBlock на поверхность разработки, выбрав элемент управления TextBlock и перетащив его на поверхность. Отцентрируйте этот элемент в верхней части окна.

Окно должно выглядеть так, как показано на следующем рисунке:



**Рис.** Окно Greetings с элементом управления TextBlock

Разметка XAML должна выглядеть приблизительно так, как показано ниже:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ae6a590c-f429-4e17-8c6b-f6fabb75ce2e');" \o "Копировать в буфер обмена.)

<TextBlock HorizontalAlignment="Center" TextWrapping="Wrap" VerticalAlignment="Center" RenderTransformOrigin="4.08,2.312" Margin="237,57,221,238"><Run Text="TextBlock"/><InlineUIContainer><TextBlock TextWrapping="Wrap" Text="TextBlock"/>

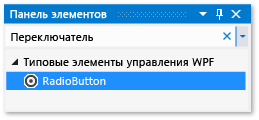
**Настройка текста в текстовом блоке**

1. В представлении XAML найдите разметку для TextBlock и измените атрибут Text: Text=”Select a message option and then choose the Display button.”
2. Если в элементе управления TextBlock не отображается весь текст в представлении конструирования, увеличьте элемент TextBlock (используя маркеры на краях).
3. Сохраните изменения, нажав клавиши Ctrl-s или с помощью меню **Файл**.

Следующий шаг — добавить в форму два элемента управления [RadioButton](https://msdn.microsoft.com/ru-ru/library/ms748363.aspx).

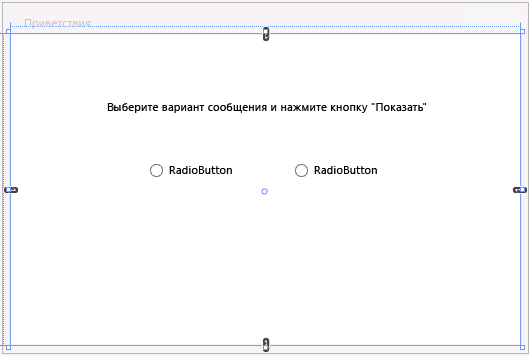
Добавление переключателей

1. На **Панели элементов** найдите элемент управления RadioButton.



1. Добавьте два элемента управления RadioButton на поверхность разработки, выбрав элемент RadioButton и дважды перетащив его на поверхность, и переместите кнопки (выбрав их и используя клавиши со стрелками), чтобы кнопки отображались рядом и под элементом управления TextBlock.

Окно должно выглядеть следующим образом:



**Рис.** Элементы управления RadioButton в окне Greetings.

1. В окне **Свойства** для левого элемента управления RadioButton измените свойство **Имя** (свойство в верхней части окна **Свойства** ), задав ему значение RadioButton1. Убедитесь, что выбран элемент управления RadioButton и в форме нет фоновой сетки; поле «Тип» в окне свойств под полем «Имя» должно иметь значение «RadioButton».
2. В окне **Свойства** для правого элемента управления RadioButton измените свойство **Имя** на RadioButton2 и сохраните изменения, нажав клавиши Ctrl-s или с помощью меню **Файл**. Перед изменением и сохранением убедитесь, что выбран элемент управления RadioButton.

Теперь можно добавить отображаемый текст для каждого элемента управления RadioButton. Следующая процедура обновляет свойство **Content** элемента управления RadioButton.

Добавление отображаемого текста для каждого переключателя

1. Откройте контекстное меню элемента управления RadioButton1 (выбрав его и нажав правую кнопку мыши), выберите команду **Изменить текст**, а затем введите Hello.
2. Откройте контекстное меню элемента управления RadioButton2 (выбрав его и нажав правую кнопку мыши), выберите команду **Изменить текст**, а затем введите Goodbye.

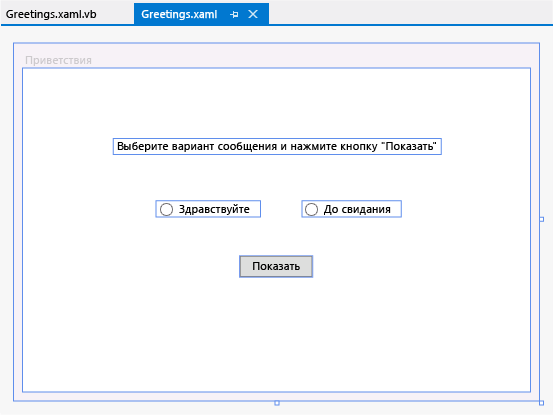
Последний элемент пользовательского интерфейса, который вам предстоит добавить, — это [Button](https://msdn.microsoft.com/ru-ru/library/ms752065.aspx).

**Добавление элемента управления Button**

1. В **Панели инструментов** найдите элемент управления **Кнопка** и добавьте его на поверхность разработки под элементами управления RadioButton, выбрав кнопку и перетащив ее на форму в представлении конструирования.
2. В представлении XAML измените значение свойства **Содержимое** элемента управления «Кнопка» с Content=”Button” на Content=”Display” и сохраните изменения (используя Ctrl-s или меню **Файл**).

Разметка должна быть аналогична разметке, приведенной в следующем примере: <Button Content="Display" HorizontalAlignment="Left" VerticalAlignment="Top" Width="75" Margin="215,204,0,0"/>

Окно должно выглядеть так, как показано на следующем рисунке.



**Рис.** Окончательный пользовательский интерфейс Greetings

**Добавление кода к кнопке Display**

После запуска приложения окно сообщения появится только тогда, когда пользователь сначала выберет переключатель, а затем нажмет кнопку **Display**. Одно окно сообщения появится для Hello, и другое — для Goodbye. Для создания такого поведения добавьте код в событие Button\_Click в файле Greetings.xaml.vb или Greetings.xaml.cs.

Добавление кода для отображения окон сообщений

1. На поверхности разработки дважды щелкните кнопку **Display**.

Будет открыт файл Greetings.xaml.vb или Greetings.xaml.cs, а курсор будет установлен в событии Button\_Click. Также можно добавить обработчик событий щелчка следующим образом (если во вставленном коде какие-либо имена подчеркнуты красной волнистой линией, то, возможно, не были выбраны и переименованы элементы управления RadioButton на поверхности разработки):

Для Visual Basic обработчик событий должен выглядеть следующим образом:

VB

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_9f3b244a-3898-484d-860d-98fae6841dcd');)

Private Sub Button\_Click\_1(sender As Object, e As RoutedEventArgs)

End Sub

Для Visual C# обработчик события должен выглядеть следующим образом:

C#

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_e0a35fba-2711-49e9-8f66-6c46d399330b');)

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

}

1. Для Visual Basic введите следующий код:

VB

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_a6ebffb4-4b1e-4994-b918-c0cd482b8d97');)

If RadioButton1.IsChecked = True Then

MessageBox.Show("Hello.")

Else RadioButton2.IsChecked = True

MessageBox.Show("Goodbye.")

End If

Для Visual C# введите следующий код:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d1bcb074-fd6a-4125-94bb-4fd9e77a5292');)

if (RadioButton1.IsChecked == true)

{

MessageBox.Show("Hello.");

}

else

{

RadioButton2.IsChecked = true;

MessageBox.Show("Goodbye.");

}

1. Сохраните приложение.

**Отладка и тестирование приложения**

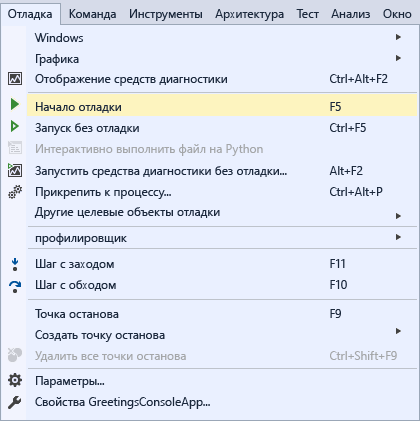
После этого вам предстоит отладить приложение для выявления ошибок и тестирования того, что оба окна сообщений отображаются правильно. Приведенные ниже инструкции описывают, как выполнить сборку и запустить отладчик .

**Поиск и исправление ошибок**

На этом шаге вам предстоит найти ошибку, которую мы намеренно допустили ранее, изменив имя файла XAML главного окна.

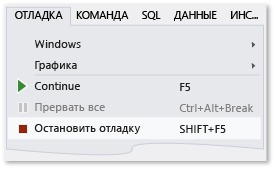
Начало отладки и поиск ошибки

1. Запустите отладчик, выбрав **Отладка**, затем **Начать отладку**.



Появится диалоговое окно, показывающее, что произошло исключение IOException "Не удалось обнаружить ресурс ‘mainwindow.xaml’".

1. Нажмите кнопку **ОК**, а затем остановите отладчик.



Файл Mainwindow.xaml был переименован в Greetings.xaml в начале этого пошагового руководства, но код по-прежнему ссылается на файл Mainwindow.xaml как на начальный универсальный код ресурса (URI) для приложения, поэтому проект не может быть запущен.

Задание Greetings.xaml в качестве начального универсального кода ресурса (URI)

1. В **Обозревателе решений** откройте файл App.xaml (в проекте C#) или файл Application.xaml (в проекте Visual Basic) в представлении XAML (его невозможно открыть в представлении конструирования), выбрав файл и нажав клавишу ВВОД или дважды щелкнув его.
2. Измените StartupUri="MainWindow.xaml" на StartupUri="Greetings.xaml" и сохраните изменения, нажав Ctrl-s.

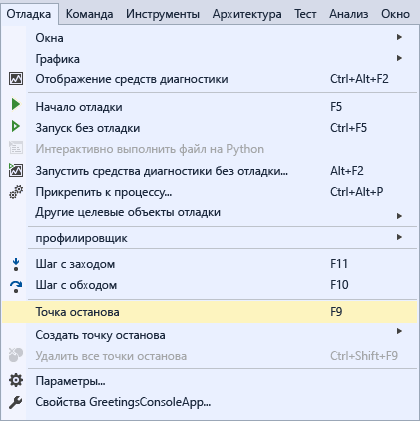
Запустите отладчик снова (клавишей F5). Должно появиться окно Greetings приложения.

**Отладка с точками останова**

Добавление некоторых точек останова позволяет тестировать код во время отладки. Для добавления точки останова можно выбрать в меню **Отладка** пункт **Точка останова** или щелкнуть в левой области редактора рядом со строкой кода, на которой требуется приостановить выполнение.

Добавление точек останова

1. Откройте файл Greetings.xaml.vb или Greetings.xaml.cs и выделите следующею строку: MessageBox.Show("Hello.")
2. Добавьте точку останова, выбрав меню **Отладка**, затем —**Точка останова**.



Рядом со строкой кода в крайнем левом поле окна редактора появится красный кружок.

1. Выделите следующую строку: MessageBox.Show("Goodbye.").
2. Нажмите клавишу F9, чтобы добавить точку останова, затем нажмите клавишу F5, чтобы начать отладку.
3. В окне **Greetings** выберите переключатель **Hello** и нажмите кнопку **Display**.

Строка MessageBox.Show("Hello.") выделяется желтым цветом. В нижней части интегрированной среды разработки окна «Видимые», «Локальные» и «Контрольные значения» закреплены вместе на левой стороне, а окна «Стек вызовов», «Точки останова», «Команда», «Интерпретация» и окно вывода закреплены вместе на правой стороне.

1. В меню **Отладка** выберите **Шаг с выходом**.

Приложение возобновит выполнение, и появится окно сообщения со словом "Hello".

1. Нажмите кнопку **ОК** в окне сообщения, чтобы закрыть его.
2. В окне **Greetings** выберите переключатель **Goodbye** и нажмите кнопку **Display**.

Строка MessageBox.Show("Goodbye.") выделяется желтым цветом.

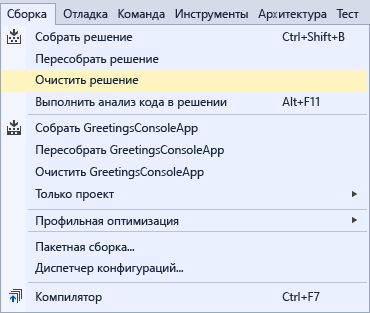
1. Нажмите клавишу F5, чтобы продолжить отладку. Когда появится окно сообщения, нажмите в нем кнопку **ОК**, чтобы закрыть его.
2. Нажмите комбинацию клавиш SHIFT + F5 (нажмите сначала клавишу SHIFT и, удерживая ее нажатой, нажмите клавишу F5), чтобы остановить отладку.
3. В меню **Отладка** выберите **Выключить все точки останова**.

**Сборка окончательной версии приложения**

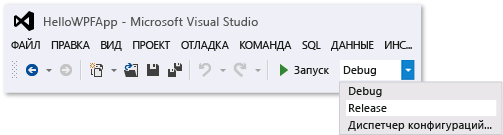
Теперь, когда вы проверили, что все работает, можно подготовить окончательную сборку приложения.

Очистка файлов решения и сборка окончательной версии

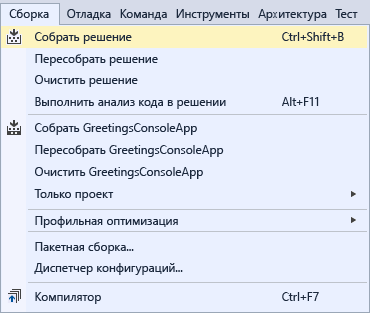
1. Выберите в главном меню **Сборка** и **Очистить решение** для удаления промежуточных файлов и выходных файлов, которые были созданы в ходе предыдущих сборок. Это не является обязательным, но очищает результаты отладочной сборки.



1. Измените конфигурацию сборки для HelloWPFApp с **Отладка** на **Выпуск** с помощью раскрывающегося списка на панели инструментов (сейчас это «Отладка»).



1. Постройте решение, выбрав **Сборка**, затем **Собрать решение** или нажмите клавишу F6.

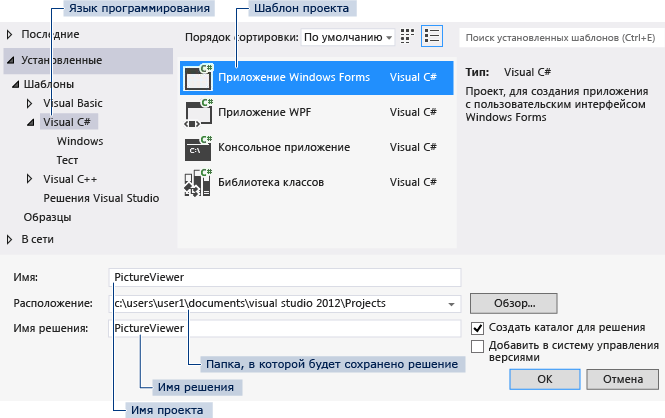


Поздравляем с завершением этого пошагового руководства! Построенный EXE-файл находится в каталоге решения и проекта (... \HelloWPFApp\HelloWPFApp\bin\Release\).

Создание проекта приложения Windows Forms

**Шаг 1. Создание проекта приложения Windows Forms**

1. В строке меню выберите **Файл**, **Создать**, **Проект**. Диалоговое окно должно выглядеть следующим образом.

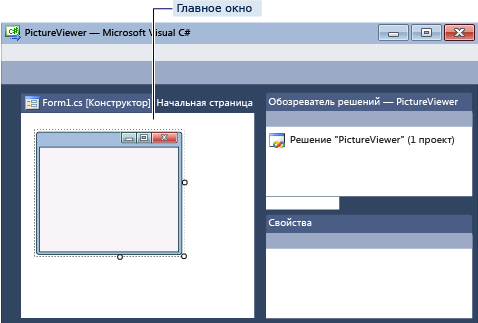
  
Диалоговое окно "Новый проект"

1. В списке **Установленные шаблоны** выберите **Visual C#** или **Visual Basic**.
2. В списке шаблонов выберите значок **Приложение Windows Forms**. Назовите новую форму PictureViewer и нажмите кнопку **ОК**.

Visual Studio создает решение для программы. Решение играет роль контейнера для всех проектов и файлов, необходимых программе.

На следующем рисунке показано, как теперь должен выглядеть интерфейс Visual Studio.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| В вашем случае макет окна может отличаться от показанного. Точный макет окна зависит от версии Visual Studio, используемого языка программирования и других факторов. Однако необходимо убедиться, что отображаются все три окна. |

1.   
   Окно интегрированной среды разработки
2. Интерфейс содержит три окна: главное окно, **Обозреватель решений** и окно **Свойства**.
3. Если какое-либо из этих окон отсутствует, восстановите макет окон по умолчанию, выбрав в строке меню **Окно**, **Сброс макета окон**. Можно также отобразить окна с помощью команд меню. В строке меню выберите **Вид**, **Окно "Свойства"** или **Обозреватель решений**. Если открыты какие-либо другие окна, закройте их с помощью кнопки **Закрыть** (x) в верхнем правом углу.
4. На рисунке показаны следующие окна (по часовой стрелке от левого верхнего угла):
   * **Главное окно** В этом окне выполняется основная часть работы, например работа с формами и редактирование кода. На рисунке в окне показана форма в редакторе форм. В верхней части окна находятся две вкладки — вкладка **Начальная страница** и вкладка **Form1.cs [Design]**. (В Visual Basic имя вкладки заканчивается на .vb, а не на .cs.)
   * **Окно Обозреватель решений** В этом окне можно просматривать все элементы, входящие в решение, и переходить к ним. Если выбрать файл, содержимое в окне **Свойства** изменится. Если открыть файл кода (с расширением .cs в Visual C# и .vb в Visual Basic), откроется файл кода или конструктор для файла кода. Конструктор — это визуальная поверхность, на которую можно добавлять элементы управления, такие как кнопки и списки. При работе с формами Visual Studio он называется конструктор Windows Forms.
   * **Окно Свойства** В этом окне производится изменение свойств элементов, выбранных в других окнах. Например, выбрав форму Form1, можно изменить ее название путем задания свойства **Text**, а также можно изменить цвет фона путем задания свойства **Backcolor**.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| В верхней строке области **Обозреватель решений** отображается текст **Решение "PictureViewer" (1 проект)**; это означает, что Visual Studio автоматически создала для вас решение. Решение может содержать несколько проектов, но пока что вы будете работать с решениями, которые содержат только один проект. |

1. В строке меню выберите **Файл**, **Сохранить все**.

Другой вариант — нажать кнопку **Сохранить все** на панели инструментов, показанной на следующем рисунке.

Кнопка "Сохранить все" на панели инструментов  
Кнопка "Сохранить все" на панели инструментов

Visual Studio автоматически заполняет имя папки и имя проекта, а затем сохраняет проект в папке проектов.

**Шаг 2. Запуск программы**

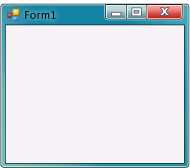
При создании нового решения фактически выполняется построение программы, которая в последующем запускается. Она пока ничего не делает — просто открывает пустое окно, у которого в строке заголовка окна надпись **Form1**. Но, как очевидно, она выполняется.

Для запуска программы

1. Для запуска программы используйте один из следующих методов.
   * Нажмите клавишу **F5**.
   * В строке меню выберите **Отладка**, **Начать отладку**.
   * На панели инструментов нажмите кнопку **Начать отладку**, которая показана на рисунке ниже.

Кнопка панели инструментов "Начать отладку"  
Кнопка панели инструментов "Начать отладку"

1. Visual Studio запускает программу, и открывается окно **Form1**. На следующей диаграмме показана только что созданная программа. Программа выполняется и скоро она будет дополнена.

  
Выполнение программы приложения Windows Form

1. Вернитесь в интегрированную среду разработки Visual Studio и посмотрите на новую панель инструментов. При запуске программы на панели инструментов появляются дополнительные кнопки. Эти кнопки позволяют выполнять такие действия, как остановка и запуск программы, а также помогают отслеживать все ошибки. В этом примере мы просто используем его для запуска и остановки программы.

Панель инструментов "Отладка"  
Панель инструментов "Отладка"

1. Для остановки программы используйте один из следующих методов.
   * На панели инструментов нажмите кнопку **Остановить отладку**.
   * В строке меню выберите **Отладка**, **Остановить отладку**.
   * Нажмите кнопку X в правом верхнем углу окна **Form1**.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Когда производится запуск программы внутри интегрированной среды разработки, это называется *отладкой*, так как это обычно выполняется для обнаружения и исправления ошибок в программе. Хотя программа очень мала и пока ничего на самом деле не делает, это все равно настоящая программа. Для запуска и отладки других программ следует выполнить ту же процедуру. |

**Шаг 3. Настройка свойств формы**

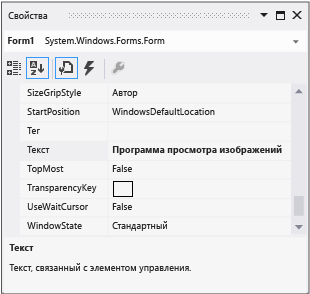
.

Далее окно **Свойства** используется для изменения внешнего вида формы.

1. Убедитесь, что вы смотрите на конструктор Windows Forms. В интегрированной среде разработки Visual Studio откройте вкладку **Form1.cs [Design]** (или вкладку **Form1.vb [Design]** в Visual Basic).
2. Чтобы выделить форму **Form1**, щелкните в любом ее месте. Посмотрите на окно **Свойства**. Теперь в нем должны отображаться свойства формы. У формы есть различные свойства. Например, можно установить цвет переднего плана и фона, текст заголовка, который отображается в верхней части формы, размер формы и другие свойства.

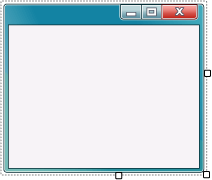
|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если окно **Свойства** не открывается, остановите программу, нажав квадратную кнопку **Остановить отладку** на панели инструментов или просто закройте окно. Если программа остановлена, но окно **Свойства** все равно не отображается, в строке меню выберите **Вид**, **Окно "Свойства"**. |

1. Когда форма будет выбрана, найдите свойство **Text** в окне **Свойства**. В зависимости от того, как отсортирован список, может потребоваться прокрутить вниз. Выберите **Text**, введите "Программа просмотра изображений", затем нажмите клавишу ВВОД. Теперь форма в заголовке окна должна содержать текст **Программа просмотра изображений**. Окно **Свойства** должно выглядеть так, как показано на рисунке ниже.

  
Окно \"Свойства\"

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Свойства можно упорядочить по категориям или в алфавитном порядке. Переключение между двумя этими представлениями можно делать с помощью кнопок в окне **Свойства**. В этом руководстве свойства легче находить в представлении, в котором свойства представлены в алфавитном порядке. |

1. Вернитесь к конструктору Windows Forms. Нажмите нижний правый маркер перетаскивания формы, который представляет собой небольшой белый квадрат в нижнем правом углу формы и показан на рисунке ниже.

  
Маркер перетаскивания

Перетащите маркер, чтобы изменить размер формы — она должна стать шире и немного выше.

1. Посмотрите в окно **Свойства** и обратите внимание, что изменилось значение свойства **Size**. Свойство **Size** меняется каждый раз при изменении формы. Перетащите маркер, чтобы форма имела размер около 550, 350 (не обязательно точно такие значения). Такой размер вполне подходит для данного проекта. В качестве альтернативы можно вводить значения непосредственно в свойстве **Size** и затем нажимать клавишу ВВОД.
2. Снова выполните программу. Помните, что можно использовать любой из следующих методов для выполнения программы.
   * Нажмите клавишу **F5**.
   * В строке меню выберите **Отладка**, **Начать отладку**.
   * На панели инструментов нажмите кнопку **Начать отладку**, которая показана на рисунке ниже.

Кнопка панели инструментов "Начать отладку"  
Кнопка панели инструментов "Начать отладку"

1. Как и ранее, интегрированная среда разработки выполняет построение программы и запускает ее, открывается окно.
2. Перед переходом к следующему шагу, остановите программу, так как интегрированная среда разработки не позволяет изменять программу при ее выполнении. Помните, что можно использовать любой из следующих методов для остановки программы.
   * На панели инструментов нажмите кнопку **Остановить отладку**.
   * В строке меню выберите **Отладка**, **Остановить отладку**.
   * Нажмите кнопку X в правом верхнем углу окна **Form1**.

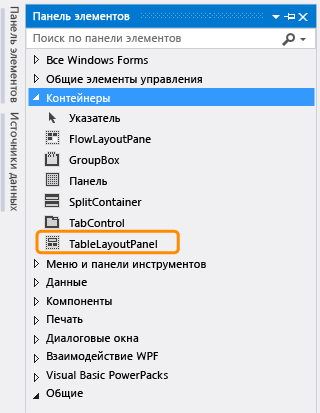
**Шаг 4. Создание макета формы с помощью элемента управления TableLayoutPanel**

.

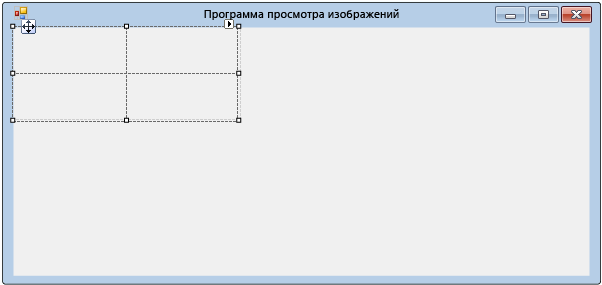
На этом шаге выполняется добавление в форму элемента управления TableLayoutPanel. TableLayoutPanel помогает должным образом выровнять элементы управления в форме, которая будет добавлена позднее.

Создание макета формы с помощью элемента управления TableLayoutPanel

1. На левой стороне среды разработки Visual Studio найдите вкладку **Панель элементов**. Выберите вкладку **Панель элементов**; появится панель элементов. (Или выберите в строке меню **Вид**, **Панель элементов**.)
2. Выберите маленький треугольник рядом с группой **Контейнеры**, чтобы открыть ее, как показано на рисунке ниже.

  
Группа "Контейнеры"

1. В форму можно добавить такие элементы управления, как кнопки, флажки и метки. Дважды щелкните элемент управления TableLayoutPanel в панели элементов. (Можно также перетащить элемент управления с панели элементов в форму.) В результате этого действия интегрированная среда разработки добавляет в форму элемент управления TableLayoutPanel, как показано на рисунке ниже.

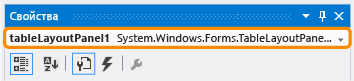
  
Элемент управления TableLayoutPanel

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| После добавления элемента управления TableLayoutPanel, если внутри формы появляется окно с заголовком **Задачи TableLayoutPanel**, чтобы закрыть его, щелкните в любом месте внутри формы. Далее в этом руководстве приводятся сведения об этом окне. |

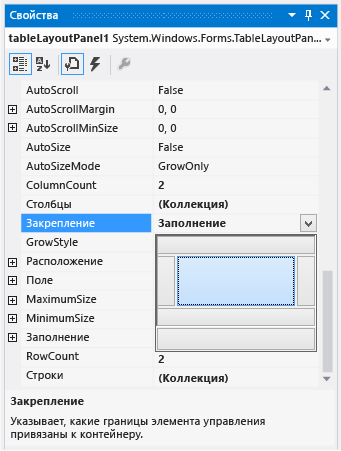
Обратите внимание, как разворачивается панель элементов, чтобы закрыть форму при нажатии на ее вкладку, и закрывается, после щелчка за ее пределами. Это функция автоматического скрытия интегрированной среды разработки. Ее можно включить или выключить для любого из окон путем нажатия значка канцелярской кнопки в правом верхнем углу окна, чтобы включить автоматическое скрытие и закрепление на месте. Появляется значок канцелярской кнопки, как показано на рисунке ниже.

**Значок канцелярской кнопки**

1. Убедитесь, что выделен элемент управления **TableLayoutPanel**, для этого щелкните по нему. Чтобы проверить, что элемент управления выделен, необходимо посмотреть в раскрывающийся список в верхней части окна **Свойства**, как показано на рисунке ниже.

  
Окно "Свойства", в котором показан элемент управления TableLayoutPanel

1. Нажмите кнопку **В алфавитном порядке** на панели инструментов в окне **Свойства**. Это приведет к отображению списка свойств в окне **Свойства** в алфавитном порядке, что упрощает поиск свойств в этом учебнике.
2. Селектор элементов управления представляет раскрывающийся список в верхней части окна **Свойства**. В данном пример он показывает, что выделен элемент управления с именем tableLayoutPanel1. Элементы управления можно выбирать, либо выделяя область в конструкторе Windows Forms, либо выбирая их в селекторе элементов управления. Теперь, когда выделен элемент управления TableLayoutPanel, найдите свойство **Dock** и выберите **Dock**, значение которого должно быть равным **None**. Обратите внимание, что рядом со значением появляется стрелка раскрывающегося списка. Нажмите стрелку, затем нажмите кнопку **Заполнение** (большая кнопка в середине), как показано на рисунке ниже.

  
Окно "Свойства", в котором нажата кнопка "Заполнение"

Под *закреплением* в Visual Studio понимается прикрепление окна к другому окну или области в интегрированной среде разработки. Например, окно "Свойства" можно открепить, т. е. отсоединить и оставить свободно плавающим в пределах Visual Studio или же закрепить в области **Обозреватель решений**.

1. После того, как у элемента управления TableLayoutPanel свойству **Dock** присвоено значение **Fill**, панель заполняет всю форму. Если снова изменить размер формы, элемент управления TableLayoutPanel останется закрепленным и сам изменит свой размер для заполнения формы.

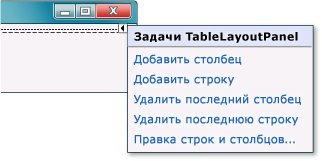
|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Элемент управления TableLayoutPanel работает как таблица в Microsoft Office Word — он содержит строки и столбцы и отдельная ячейка может занимать несколько строк и столбцов. Каждая ячейка может содержать один элемент управления (например, кнопку, флажок или метку). Этот элемент управления TableLayoutPanel будет содержать элемент управления PictureBox, который займет всю верхнюю строку, элемент управления CheckBox в левой нижней ячейке, и четыре элемента управления Button в правой нижней ячейке. |

1. В данный момент элемент управления TableLayoutPanel содержит две одинаковые по размеру строки и два одинаковых по размеру столбца. Нужно изменить их размер, чтобы верхняя строка и правый столбец были намного больше. В конструкторе Windows Forms выберите элемент управления TableLayoutPanel. В правом верхнем углу расположена маленькая кнопка с черным треугольником, как показано на рисунке ниже.

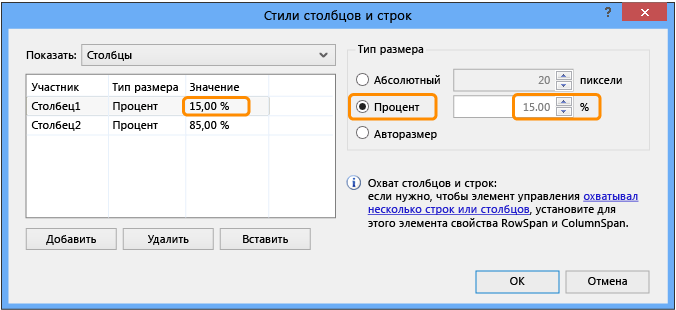
Кнопка с треугольником  
Кнопка с треугольником

Это кнопка указывает, что элемент управления содержит задачи, которые помогут автоматически задать его свойства.

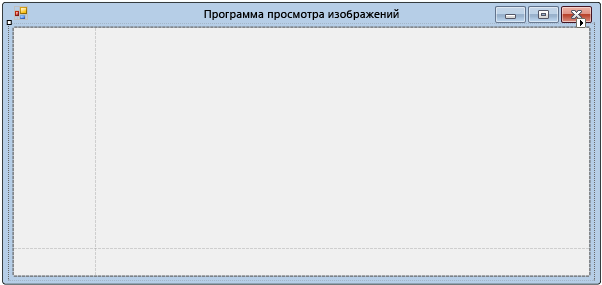
1. Чтобы отобразить список задач элемента управления, как показано на рисунке ниже, нажмите треугольник.

  
Задачи элемента управления TableLayoutPanel

1. Выберите задачу **Изменить строки и столбцы**, чтобы открыть окно **Стили столбцов и строк**. Выберите **Столбец1**, убедитесь, что нажата кнопка **Проценты**, установите его размер равным 15 процентам — введите 15 в поле **Проценты**. (это элемент управления NumericUpDown, который далее будет использоваться в этом руководстве). Выберите **Столбец2** и задайте значение 85 процентов. Пока не нажимайте кнопку **ОК**, так как окно будет закрыто. (но если это было сделано, его можно открыть повторно с помощью списка задач).

  
Стили столбцов и строк TableLayoutPanel

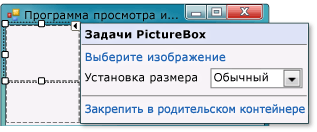
1. В раскрывающемся списке **Показать** в верхней части окна выберите **Строки**. Задайте для **Row1** значение 90 процентов, а для **Row2** 10 процентов.
2. Нажмите кнопку **ОК**. Элемент управления TableLayoutPanel теперь должен содержать большую верхнюю строку, маленькую нижнюю строку, маленький левый столбец и большой правый столбец. Можно изменить размер строк и столбцов в элементе управления TableLayoutPanel, выбрав tableLayoutPanel1 в форме, а затем перетащив границы его строк и столбцов.

  
Form1 с измененным размером TableLayoutPanel

# **Шаг 5. Добавление элементов управления в форму**

На данном шаге производится добавление на форму элемента управления PictureBox и элемента управления CheckBox. Затем на форму добавляются кнопки.

1. Перейдите на вкладку "Панель элементов" (находящуюся в левой части интегрированной среды разработки Visual Studio) и разверните группу **Стандартные элементы управления**. В результате этого действия отображается большая часть стандартных элементов управления, которые можно увидеть в формах.
2. Выберите элемент управления TableLayoutPanel в форме. Чтобы подтвердить, что элемент TableLayoutPanel выбран, убедитесь, что его имя отображается в раскрывающемся списке в верхней части окна **Свойства**. Выбирать элементы управления в форме можно также с помощью раскрывающегося списка в верхней части окна **Свойства**. Выбирать элементы управления таким образом зачастую проще, чем выбрать крошечный элемент управления с помощью мыши.
3. Дважды щелкните элемент **PictureBox**, чтобы добавить в форму элемент управления PictureBox. Поскольку элемент управления TableLayoutPanel закреплен так, чтобы заполнять собой форму, интегрированная среда разработки добавляет элемент управления PictureBox в первую пустую ячейку (левый верхний угол).
4. Щелкните новый элемент управления PictureBox, чтобы выбрать его, а затем щелкните черный треугольник на новом элементе управления PictureBox, чтобы отобразить его список задач, как показано на следующем рисунке.

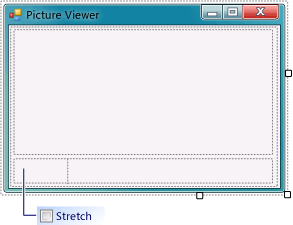
  
Задачи элемента управления PictureBox

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если на TableLayoutPanel случайно был добавлен элемент управления неправильного типа, то его можно удалить. Щелкните элемент управления правой кнопкой мыши и в контекстном меню выберите **Удалить**. Удалять элементы управления из формы также можно с помощью строки меню. В строке меню выберите **Правка**, **Отменить** или **Правка**, **Удалить**. |

1. Выберите ссылку **Закрепить в родительском контейнере**. В результате этого действия у элемента управления PictureBox свойство **Dock** принимает значение **Fill**. Чтобы это увидеть, выберите элемент управления PictureBox, перейдите к окну **Свойства** и убедитесь, что свойство **Dock** имеет значение **Fill**.
2. Сделайте так, чтобы элемент управления PictureBox занимал два столбца с помощью его свойства **ColumnSpan**. Выберите элемент управления PictureBox и установите для его свойства **ColumnSpan** значение **2**. Также необходимо, чтобы когда элемент управления PictureBox был пустым, отображалась пустая рамка. Установите для его свойства **BorderStyle** значение **Fixed3D**.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если у PictureBox отсутствует свойство **ColumnSpan**, вероятно, элемент управления PictureBox был добавлен в саму форму, а не в элемент управления TableLayoutPanel. Чтобы исправить это, выберите элемент управления PictureBox, удалите его, выберите элемент TableLayoutPanel, а затем добавьте новый элемент управления PictureBox. |

1. Выберите элемент управления TableLayoutPanel в форме и добавьте в форму элемент управления **CheckBox**. Двойным щелчком выберите элемент **CheckBox** на панели элементов, чтобы добавить новый элемент управления CheckBox в следующую свободную ячейку таблицы. Поскольку элемент управления PictureBox занимает первые две ячейки в TableLayoutPanel, элемент управления CheckBox добавляется в нижнюю левую ячейку. Выберите свойство **Text** и введите слово Stretch, как показано на следующем рисунке.

  
Элемент управления TextBox со свойством Stretch

1. Выберите в форме элемент управления TableLayoutPanel, а затем на панели элементов перейдите к группе **Контейнеры** (из которой был взят элемент управления TableLayoutPanel) и дважды щелкните элемент управления **FlowLayoutPanel**, чтобы добавить новый элемент управления в последнюю ячейку в элементе управления PictureBox (справа внизу). Затем закрепите FlowLayoutPanel в TableLayoutPanel (выбрав **Закрепить в родительском контейнере** в списке задач FlowLayoutPanel, раскрываемом с помощью черного треугольника, или установив свойству **Dock** FlowLayoutPanel значение **Fill**).

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Элемент управления FlowLayoutPanel является контейнером, который размещает другие элементы управления аккуратно по строкам и в определенном порядке. Когда изменяется размер элемента управления FlowLayoutPanel, если он содержит достаточно места для размещения всех компонентов в одной строке, то он размещает их именно таким образом. В противном случае он размещает их по строкам одну над другой. Элемент управления FlowLayoutPanel будет использоваться для размещения четырех кнопок. Если кнопки при добавлении располагаются одна поверх другой, убедитесь, что перед добавлением кнопок вы выбрали FlowLayoutPanel. Хотя ранее было сказано, что каждая ячейка может содержать только один элемент управления, нижняя правая ячейка TableLayoutPanel содержит четыре элемента управления "Кнопка". Это так, потому что можно разместить элемент управления в ячейке, которая содержит другие элементы управления. Такой тип элементов управления называется "контейнер", и элемент управления FlowLayoutPanel является контейнером. |

**Добавление кнопок**

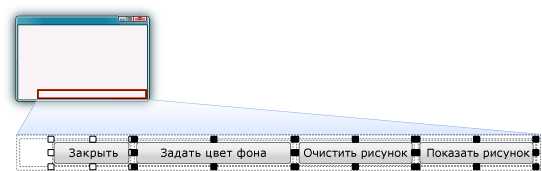
1. Выберите только что добавленный элемент управления FlowLayoutPanel. Перейдите к группе **Стандартные элементы управления** на панели элементов и двойным щелчком выберите элемент **Кнопка**, чтобы добавить в элемент управления FlowLayoutPanel кнопку с именем **button1**. Чтобы добавить другую кнопку, повторите это действие. Среда интегрированной разработки определяет, что уже существует кнопка с именем **button1** и называет следующую кнопку как **button2**.
2. Обычно другие кнопки добавляются при помощи панели элементов. На этот раз выберите **button2**, затем в строке меню выберите **Правка**, **Копировать** (или нажмите сочетание клавиш CTRL+C). В строке меню выберите **Правка**, **Вставить** (или нажмите сочетание клавиш CTRL+V), чтобы вставить копию кнопки. Повторите вставку еще раз. Интегрированная среда разработки добавила кнопки **button3** и **button4** в FlowLayoutPanel.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Любой элемент управления можно копировать и вставлять. Среда интегрированной разработки именует и размещает новые элементы управления логическим образом. Если вставка элемента управления выполняется в контейнер, среда интегрированной разработки выбирает следующую логическую область для размещения. |

1. Выберите первую кнопку и установите для ее свойства **Text** значение "Показать рисунок". Затем установите для свойства **Text** следующих трех кнопок значения "Очистить рисунок", "Установить цвет фона" и "Закрыть".
2. Следующий шаг — задать размер кнопок и разместить их таким образом, чтобы они были выровнены по правой стороне панели. Выберите элемент управления FlowLayoutPanel и обратите внимание на его свойство **FlowDirection**. Измените его значение на **RightToLeft**. После этого действия кнопки должны сами выровняться по правой стороне ячейки и изменить свой порядок таким образом, чтобы кнопка **Показать рисунок** располагалась с правой стороны.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если кнопки по-прежнему остаются в неправильном порядке, можно перетащить кнопки вокруг элемента управления FlowLayoutPanel для расположения их в произвольном порядке. Можно выбрать кнопку и перетащить ее влево или вправо. |

1. Щелкните кнопку **Закрыть**, чтобы выбрать ее. Удерживая клавишу CTRL, щелкните три другие кнопки, чтобы все они были выбраны. При выделенных кнопках перейдите к окну **Свойства** и прокрутите его вверх до свойства **AutoSize**. Это свойство указывает кнопке автоматически изменять свой размер так, чтобы весь текст мог разместиться на ней. Задайте значение **true**. Кнопки теперь должны иметь соответствующий размер и быть расположены в правильном порядке. (пока выделены все четыре кнопки, можно одновременно изменить все четыре свойства **AutoSize**). На следующем рисунке показаны эти четыре кнопки.

  
Программа просмотра изображений с четырьмя кнопками

1. Теперь снова запустите программу, чтобы увидеть обновленную компоновку формы. При нажатии кнопок и установке флажка пока ничего не происходит, однако вскоре все заработает.

**Шаг 6. Присвоение имен элементам управления "Кнопка"**

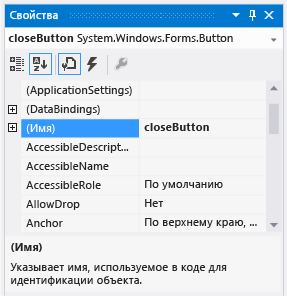
В форме существует только один элемент управления PictureBox. Когда он был добавлен, интегрированная среда разработки автоматически присвоила ему имя **pictureBox1**. Существует только один элемент управления CheckBox с именем **checkBox1**. Скоро будет написан некоторый код. В этом коде будет обращение к элементам управления CheckBox и PictureBox. Так как существуют только по одному экземпляру каждого компонента, то становится ясно, что означает упоминание имен **pictureBox1** или **checkBox1** в коде.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| В Visual Basic по умолчанию первая буква любого имени элемента управления является заглавной, поэтому у элементов управления имена **PictureBox1**, **CheckBox1** и так далее. |

В форме есть четыре кнопки. Интегрированная среда разработки назвала их как **button1**, **button2**, **button3** и **button4**. Только по их текущему имени нельзя узнать, какая кнопка является кнопкой **Закрыть**, а какая кнопкой **Показать рисунок**. Вот почему присвоение элементам управления в виде кнопок более осмысленных названий полезно.

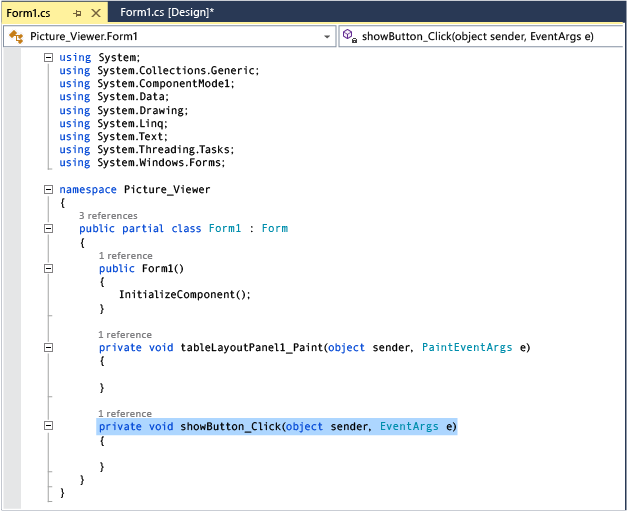
**Присвоение имен элементам управления "Кнопка"**

1. В форме нажмите кнопку **Закрыть**. (Если все еще выделены все кнопки, для отмены выделения нажмите клавишу ESC.) Прокрутите содержимое окна **Свойства**, пока не появиться свойство **(Name)**. Свойство **(Name)** расположено в верхней части, когда свойства расположены в алфавитном порядке. Измените имя на **closeButton**, как показано на рисунке ниже.

  
Окно "Свойства" с именем closeButton

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если попробовать изменить имя кнопки на **closeButton**, с пробелом между словами close и Button, среда интегрированной разработки выведет сообщение об ошибке «Недопустимое значение свойства». Пробелы (а также несколько других символов) запрещено использовать в именах элементов управления. |

1. Переименуйте другие три кнопки как **backgroundButton**, **clearButton**, **showButton**. Имена можно проверить в раскрывающемся списке селектора элементов управления в окне **Свойства**. Отобразятся новые имена кнопок.
2. Двойным щелчком нажмите кнопку **Показать рисунок** в форме. В качестве альтернативы можно нажать кнопку **Показать рисунок** в форме, а затем нажать клавишу ВВОД. При этом интегрированная среда разработки открывает в главном окне дополнительную вкладку, которая называется **Form1.cs** (**Form1.vb**, если используется Visual Basic). На этой вкладке отображается файл кода для формы, как показано на следующем рисунке.

  
Вкладка Form1.cs с кодом Visual C#

1. Обратите внимание на эту часть кода

private void showButton\_Click(object sender, EventArgs e)

{

}

Вы видите код с именем showButton\_Click(). Интегрированная среда разработки добавила его в код формы при открытии файла кода для кнопки **showButton**. Во время разработки при открытии файла кода для элемента управления в форме для элемента управления создается код, если он еще не существует. Этот код, известный как *метод*, выполняется при запуске программы и использовании этого элемента управления (в данном случае — кнопка **Показать рисунок**).

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| В этом руководстве код Visual Basic, который создается автоматически, был упрощен путем удаления всего, что находилось в круглых скобках (). Когда это происходит, одинаковый код можно удалить. Программа будет работать в любом случае. В оставшейся части руководства весь автоматически созданный код был упрощен везде, где это было возможным. |

1. Снова выберите вкладку конструктора Windows Forms (**Form1.cs [Design]** в Visual C#, **Form1.vb [Design]** в Visual Basic), а затем откройте файл кода для кнопки **Очистить рисунок**, чтобы создать метод для нее в коде формы. Повторите это действие для двух оставшихся кнопок. Каждый раз при этом действии среда интегрированной разработки добавляет в файл кода формы новый метод.
2. Чтобы добавить еще один метод, откройте файл кода для элемента управления CheckBox в конструкторе Windows Forms, чтобы интегрированная среда разработки создала метод checkBox1\_CheckedChanged(). Этот метод вызывается каждый раз, когда пользователь устанавливает или снимает флажок.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| При работе с программой необходимо часто переключаться между редактором кода и конструктором Windows Forms. Среда интегрированной разработки упрощает передвижение по проекту. Используйте **Обозреватель решений**, чтобы открыть конструктор Windows Forms с помощью двойного щелчка по **Form1.cs** в Visual C# или **Form1.vb** в Visual Basic; или выберите **Вид**, **Конструктор** в строке меню. |

1. Ниже показан новый код, который представлен в редакторе кода.

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_859cdc5e-f503-4b2c-9fa3-f5f79b95e920');)

private void clearButton\_Click(object sender, EventArgs e)

{

}

private void backgroundButton\_Click(object sender, EventArgs e)

{

}

private void closeButton\_Click(object sender, EventArgs e)

{

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

}

1. Пять методов, которые были добавлены, называются *обработчики событий*, так как программа вызывает их каждый раз, когда происходит событие (например, пользователь нажимает кнопку или устанавливает флажок).
2. При просмотре кода для элемента управления в интегрированной среде разработки во время разработки Visual Studio добавляет метод обработчика событий для элемента управления, если он не существует. Например, при двойном щелчке по кнопке интегрированная среда разработки добавляет обработчик события Click, который вызывается каждый раз, когда пользователь нажимает кнопку. Если дважды щелкнуть флажок, интегрированная среда разработки добавляет обработчик события CheckedChanged, который вызывается каждый раз, когда пользователь устанавливает или снимает флажок.
3. После добавления обработчика событий для элемента управления к нему можно вернуться в любой момент из конструктора Windows Forms с помощью двойного щелчка по элементу управления или путем выбора пунктов **Вид**, **Код** в строке меню.
4. Имена являются важными при выполнении построения программы, и методы (включая обработчики событий) могут иметь любые имена, которые нужны. При добавлении обработчика событий с помощью интегрированной среды разработки она создает имя на основе имени элемента управления и обрабатываемого события. Например, событие Click для кнопки с именем **showButton** вызывает метод обработчика событий showButton\_Click(). Также обычно после имени метода добавляются открывающая и закрывающая круглые скобки () для индикации того, какие методы рассматриваются. Если вы примете решение изменить имя переменной кода, щелкните правой кнопкой мыши по переменной в коде, а затем выберите команду **Рефакторинг**, **Переименовать**. Все экземпляры этой переменной в коде будут переименованы.

**Шаг 7. Добавление компонентов Dialog в форму**

Чтобы программа могла открывать графические файлы и изменять цвет фона, на данном шаге следует добавить в форму компоненты **OpenFileDialog** и **ColorDialog**.

В некотором смысле компонент похож на элемент управления. Для добавления компонента в форму используется панель элементов, его свойства настраиваются в окне **Свойства**. Но в отличие от элемента управления, добавление в форму компонента не добавляет в форму элемент, который может видеть пользователь. Вместо этого, компонент предоставляет определенное поведение, которое можно включать в коде. Имеется компонент, который открывает диалоговое окно **Открыть файл**.

1. Выберите конструктор Windows Forms (Form1.cs [Design] или Form1.vb [Design]) и откройте группу **Диалоговые окна** в панели элементов.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Группа **Диалоговые окна** в панели элементов содержит компоненты, которые открывают множество полезных диалоговых окон. Эти диалоговые окна могут использоваться для открытия и сохранения файлов, просмотра папок, выбора шрифтов и цветов. В этом проекте используется два компонента диалоговых окон — **OpenFileDialog** и **ColorDialog**. |

1. Чтобы добавить в форму компонент с именем **openFileDialog1** дважды щелкните компонент **OpenFileDialog**. Для добавления в форму компонента с именем **colorDialog1** дважды щелкните в панели элементов компонент **ColorDialog**. (такой компонент используется в следующем шаге руководства). В нижней части конструктора Windows Forms должно быть поле (под формой программы просмотра изображений), которое содержит значок для каждого из двух добавленных компонентов диалоговых окон, как показано на рисунке ниже.

Компоненты диалоговых окон  
Компоненты диалоговых окон

1. Выберите значок **openFileDialog1** в области в нижней части конструктора Windows Forms. Установите значения двух свойств.
   * Задайте свойству **Фильтр** следующее значение (можно копировать и вставить):

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_002c0f4c-5d69-4a2b-824f-c413f7a9c789');)

JPEG Files (\*.jpg)|\*.jpg|PNG Files (\*.png)|\*.png|BMP Files (\*.bmp)|\*.bmp|All files (\*.\*)|\*.\*

* + Установите для свойства **Title** следующее значение "Выбор файла изображения".

Параметры свойства **Фильтр** определяют типы файлов, которые отображаются в диалоговом окне **Выбор файла изображения**.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Чтобы посмотреть пример диалогового окна **Открыть файл** в другом приложении, откройте программу "Блокнот" или Paint и в меню **Файл** выберите пункт **Открыть**. Обратите внимание на раскрывающийся список **Типы файлов** в нижней части окна. Только что было настроено свойство **Filter** у компонента **OpenFileDialog**. Также, обратите внимание как выделены полужирным шрифтом свойства **Title** и **Filter** в окне **Свойства**. Таким образом интегрированная среда разработки показывает свойства, у которых были изменены их значения по умолчанию. |

**Шаг 8. Написание кода для обработчика событий кнопки "Показать рисунок"**

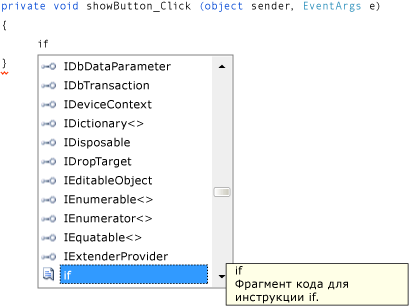
На этом шаге выполняется создание кнопки **Показать рисунок**, которая работает следующим образом.

* Когда пользователь нажимает эту кнопку, программа открывает диалоговое окно **Открыть файл**.
* Если пользователь открывает файл рисунка, программа показывает этот рисунок в PictureBox.

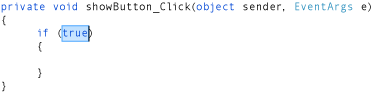
Среда IDE содержит мощное средство IntelliSense, которое помогает в написании кода. По мере написания кода, среда IDE открывает поле, в котором содержатся предлагаемые завершения для частей вводимых слов. Она пытается определить, что требуется сделать далее и автоматически переходит к последнему выбранному элементу из списка. Для перемещения по списку можно использовать клавиши со стрелками ВВЕРХ или ВНИЗ или можно продолжать вводить буквы, чтобы сузить выбор. Когда появится необходимый элемент, для его выбора нажмите клавишу TAB. Либо можно проигнорировать предложение, если оно не требуется.

**Написание кода для обработчика событий кнопки "Показать рисунок"**

1. Перейдите к конструктору Windows Forms и дважды щелкните кнопку **Показать рисунок**. Среда интегрированной разработки немедленно переключается на конструктор кода и перемещает курсор внутрь метода showButton\_Click(), который был добавлен ранее.
2. Введите i в пустой строке между двумя фигурными скобками { } (в Visual Basic введите пустую строку между Private Sub… and End Sub). Откроется окно **IntelliSense**, как показано на рисунке ниже.

  
IntelliSense с кодом Visual C#

1. Окно **IntelliSense** должно выделить слово **if**. (в противном случае введите в нижнем регистре f для этого). Обратите внимание, каким образом небольшое окно *подсказка* рядом с окном **IntelliSense** отображается с описанием **Фрагмент кода для инструкции if**. (в Visual Basic подсказка также указывает такой фрагмент, но немного с другим содержимым). Необходимо использовать этот фрагмент, поэтому нажмите клавишу TAB, чтобы вставить **if** в свой код. Затем снова нажмите клавишу TAB, чтобы использовать фрагмент **if**. (Если вы выберете что-то другое и окно **IntelliSense** исчезнет, нажмите клавишу BACKSPACE, чтобы удалить **i**, и повторно введите эту букву, чтобы снова открыть окно **IntelliSense**).

  
Код Visual C#

1. Далее IntelliSense используется для ввода дополнительного кода для открытия диалогового окна **Открыть файл**. Если пользователь нажимает кнопку **ОК**, PictureBox загружает выбранный пользователем файл. Следующие действия показывают как ввести код. Хотя представлено множество действий, это просто несколько нажатий клавиш.
   1. Начните с выделенным текстом **true** в фрагменте. Введите op, чтобы перезаписать его. (в Visual Basic необходимо начинать с первой заглавной буквы, поэтому введите Op).
   2. Откроется окно **IntelliSense** и отобразит **openFileDialog1**. Нажмите клавишу TAB, чтобы выбрать его. (в Visual Basic он начинается с заглавной буквы, поэтому будет представлен **OpenFileDialog1**; убедитесь, что выделен **OpenFileDialog1**).

Дополнительные сведения о OpenFileDialog см. в разделе [OpenFileDialog](https://msdn.microsoft.com/library/system.windows.forms.openfiledialog.aspx).

* 1. Введите точку (.). Так как точка введена справа после элемента **openFileDialog1**, окно **IntelliSense** открывается с методами и свойствами компонента **OpenFileDialog**. Это те же самые свойства, которые отображаются в окне **Свойства** при выборе этого окна в конструкторе Windows Forms. Также можно выбрать методы, которые дают указания компонентам на выполнение определенных действий (например, открыть диалоговое окно).

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Окно **IntelliSense** может показывать свойства и методы. Чтобы определить, какие элементы отображаются, проверьте значок слева от каждого элемента в окне **IntelliSense**. Рядом с каждым методом представлен значок кубика, рядом с каждым свойством представлен значок гаечного ключа. Также рядом с каждым событием представлен значок с изображением молнии. Ниже представлены эти значки. |

* 1. Значок метода  
     Значок метода
  2. Значок свойства  
     Значок свойства
  3. Значок события  
     Значок события
  4. Начните набирать ShowDialog (для IntelliSense регистр значения не имеет). Метод ShowDialog() будет открывать диалоговое окно **Открыть файл**. После выделения **ShowDialog** в окне нажмите клавишу TAB. Также можно выделить "ShowDialog" и нажать клавишу F1 для получения соответствующей справки.

Дополнительные сведения о методе ShowDialog() см. в разделе [Метод ShowDialog](https://msdn.microsoft.com/library/c7ykbedk.aspx).

* 1. При использовании метода в элементе управления или компоненте (такое использование называется *вызов метода*) необходимо добавить круглые скобки. Введите открывающую и закрывающую скобки сразу после "g" в разделе ShowDialog: (); теперь это должно принять вид "openFileDialog1.ShowDialog()".

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Методы являются важнейшей частью любой программы. В этом руководстве показано несколько способов использования методов. Можно вызвать метод компонента, чтобы указать ему выполнение некоторых действий, например, как у компонента **OpenFileDialog** вызывается метод ShowDialog(). Можно создать собственные методы, чтобы программа выполняла определенные действия, например метод, создание которого выполняется сейчас, выполняла вызов метода showButton\_Click(), который открывает диалоговое окно и рисунок при нажатием пользователем кнопки. |

* 1. В Visual C# добавьте пробел, затем два знака равенства (==). В Visual Basic добавьте пробел, затем один знак равенства (=). В Visual C# и в Visual Basic используются разные операторы равенства.
  2. Добавьте еще один пробел. Как только это будет сделано, откроется другое окно **IntelliSense**. Начните вводить DialogResult и нажмите клавишу TAB, чтобы добавить его.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| При написании кода для вызова метода, в некоторых случаях он возвращает значение. В данном случае у компонента **OpenFileDialog** метод ShowDialog() возвращает значение DialogResult. DialogResult — это специальное значение, которое указывает на событие, которое происходит в диалогом окне. В компоненте **OpenFileDialog** пользователь может нажать кнопку **ОК** или **Отмена**, чтобы метод ShowDialog() возвращал значение DialogResult.OK или значение DialogResult.Cancel. |

* 1. Чтобы открыть значение DialogResult в окне **IntelliSense** введите точку. Введите букву O и нажмите клавишу TAB, чтобы вставить **ОК**.

Дополнительные сведения о DialogResult см. в разделе [DialogResult](https://msdn.microsoft.com/library/system.windows.forms.dialogresult.aspx).

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Первая строка кода должна быть завершена. В Visual C# это выглядит следующим образом.  if (openFileDialog1.ShowDialog() == DialogResult.OK)  В Visual Basic это выглядит следующим образом.  If OpenFileDialog1.ShowDialog() = DialogResult.OK Then |

* 1. Теперь добавьте несколько строк кода. Их можно ввести вручную (или копировать и вставить), однако попробуйте использовать для добавления строк IntelliSense. Чем больше вы знакомы с IntelliSense, тем быстрее можете писать собственный код. Итоговая реализация метода showButton\_Click() будет выглядеть следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d8c7a484-8ad6-4687-8193-93c2bb2fa472');)

private void showButton\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

pictureBox1.Load(openFileDialog1.FileName);

}

}

**Шаг 9. Проверка, комментирование и тестирование кода**

Затем в код добавляется комментарий. Комментарий это заметка, которая не влияет на выполнение программы. Он облегчает понимание назначения кода тем, кто читает этот код. Добавление комментариев в код — хорошая привычка. В Visual C# строка обозначается как комментарий с помощью двух косых черт (//). В Visual Basic строка обозначается как комментарий с помощью одинарной кавычки('). После добавления комментария программа проверяется. Рекомендуется регулярно выполнять и тестировать код при работе над проектами, чтобы можно было на ранних стадиях выявлять и устранять проблемы, прежде чем код станет слишком сложным. Это называется *итеративным тестированием*.

Только что выполнено построение программы, которая работает. Хотя программа еще не завершена, но она уже может загружать рисунок. Перед добавлением комментариев в код и его проверкой, уделите время на ознакомление с понятиями кода, так как эти понятия будут часто использоваться.

* При двойном щелчке по кнопке **Показать рисунок** в конструкторе Windows Forms интегрированная среда разработки автоматически добавляла *метод* в код программы.
* Методами называют порядок организации кода — это то, каким образом группируется код.
* Большую часть времени метод выполняет небольшое количество действий в определенном порядке, например, метод showButton\_Click() показывает диалоговое окно и затем загружает рисунок.
* Метод состоит из *операторов* кода или строк кода. Считайте, что метод — это способ объединения инструкций кода.
* Когда метод выполняется или *вызывается*, операторы в методе выполняются по порядку, один за другим, начиная с первого.

Ниже приведен пример оператора.

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_da15814e-5212-4b2a-aef6-d69be79460ad');)

pictureBox1.Load(openFileDialog1.FileName);

Операторы это то, что указывает программам выполнять действия. В Visual C# оператор всегда заканчивается точкой с запятой. В Visual Basic конец строки это конец оператора. (в Visual Basic точка с запятой не нужна). Предыдущий оператор указывал элементу управления PictureBox загрузить файл, который пользователь выбрал в компоненте **OpenFileDialog**.

**Добавление комментариев**

1. Добавьте следующий комментарий в код.

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_abfc8a5c-e62b-4708-92e9-407db5ee7597');)

private void showButton\_Click(object sender, EventArgs e)

{

// Show the Open File dialog. If the user clicks OK, load the

// picture that the user chose.

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

pictureBox1.Load(openFileDialog1.FileName);

}

}

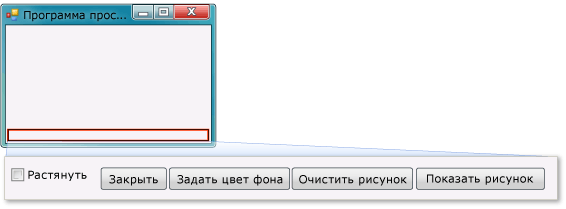
|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Обработчик события Click кнопки **showButton** завершен и он работает. Написание кода было начато с оператора if. Оператор if как-будто говорит программе: "Проверь это условие. Если оно выполняется, выполни эти действия". В данном случае программе дается указание открыть диалоговое окно **Открыть файл** и, если пользователь выбирает файл и нажимает кнопку **ОК**, загрузить этот файл в элемент управления PictureBox. |
| **System_CAPS_ICON_tip.jpg Совет** |
| Интегрированная среда разработки построена так, чтобы облегчить написание кода и *фрагменты кода* являются одним из этих способов. Фрагмент представляет собой ярлык, который разворачивается в небольшой блок кода.  Можно просмотреть все доступные фрагменты. В строке меню выберите пункт **Сервис**, **Диспетчер фрагментов кода**. В случае Visual C# фрагмент if находится в разделе **Visual C #**. В случае Visual Basic фрагменты if находятся в разделе **Условные выражения и циклы**, **Шаблоны кода**. Этот диспетчер можно использовать для просмотра существующих фрагментов и для добавления собственных фрагментов.  Чтобы активировать фрагмент при наборе кода, введите его и нажмите клавишу TAB. В окне **IntelliSense** появится множество фрагментов, поэтому нажмите клавишу TAB дважды — первый раз, чтобы выделить фрагмент в окне **IntelliSense**, второй раз, чтобы дать указание интегрированной среде разработки на использование этого фрагмента. (IntelliSense поддерживает фрагмент if, но не фрагмент ifelse). |

1. Перед выполнением программы сохраните программу, нажав кнопку панели инструментов **Сохранить все**, которая показана на рисунке ниже.

Кнопка "Сохранить все" на панели инструментов  
Кнопка "Сохранить все"

Кроме того, чтобы сохранить программу, в строке меню выберите **Файл**, **Сохранить все**. Рекомендуется выполнять сохранение от начала разработки и как можно чаще.

При выполнении программа должна выглядеть, как показано на рисунке ниже.



**Программа просмотра изображений**

**Проверка программы**

1. Нажмите клавишу F5 или нажмите кнопку **Начать отладку** на панели инструментов.
2. Чтобы выполнить только что написанный вами код, нажмите кнопку **Показать рисунок**. Сначала программа открывает диалоговое окно **Открыть файл**. Проверьте, что в нижней части диалогового окна в раскрывающемся списке **Типы файлов** появились фильтры. Затем перейдите к рисунку и откройте его. Обычно образцы рисунков, которые поставляются вместе с операционной системой Windows, можно найти в папке **Мои документы** во вложенной папке **Мои рисунки\Образцы рисунков**.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Если вы не видите никаких изображений в диалоговом окне **Выбор файла изображения**, убедитесь, что в раскрывающемся списке в нижней правой части диалогового окна выбран фильтр "Все файлы (\*.\*)". |

1. Загрузите рисунок и он появится в элемент управления PictureBox. Затем снова попробуйте изменить размер формы, перетащив ее границы. Так как элемент управления PictureBox закреплен внутри элемента управления TableLayoutPanel, который сам закреплен внутри формы, область картинки будет сама изменять размер, таким образом, что по ширине она будет как форма и заполнит 90 процентов формы. Поэтому используются контейнеры TableLayoutPanel и FlowLayoutPanel — они сохраняют правильные размеры формы, когда пользователь меняет ее размер.

Сейчас большие изображения выходят за границы средства просмотра изображений. В следующем шаге вам предстоит добавить код, чтобы изображения помещались в окне.

**Шаг 10. Написание кода дополнительных кнопок и флажка**

Теперь можно завершить другие четыре метода. Можно копировать и вставить этот код, но для получения дополнительных навыков введите код и используйте IntelliSense.

Этот код добавляет функциональность к ранее добавленным кнопкам. Без этого кода кнопки ничего не делают. Код используется в событиях Click кнопок (в случае флажка это событие CheckChanged) для выполнения различных действий при активации пользователем этих элементов управления. Например, событие clearButton\_Click, которое активируется при нажатии кнопки **Очистить рисунок**, удаляет текущее изображение, установив его свойству Image значение null (или nothing). Каждое событие в коде сопровождается комментариями, которые поясняют, что делает код.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Рекомендация — всегда снабжайте код комментариями. Комментарии — это сведения для человека, который читает код, необходимы для того, чтобы сделать код понятным. Содержимое в строке комментария игнорируется программой. В Visual C# строка комментария начинается с двух символов косой черты (//), в Visual Basic строка комментария начинается с одного знака одинарной кавычки ('). |

Написание кода для дополнительных кнопок и флажка

* Добавьте следующий код в файл кода Form1 (Form1.cs или Form1.vb). [Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f65d63f3-afb8-4ce5-bba3-8788200be287');)

Private Sub clearButton\_Click() Handles clearButton.Click

' Clear the picture.

PictureBox1.Image = Nothing

End Sub

Private Sub backgroundButton\_Click() Handles backgroundButton.Click

' Show the color dialog box. If the user clicks OK, change the

' PictureBox control's background to the color the user chose.

If ColorDialog1.ShowDialog() = DialogResult.OK Then

PictureBox1.BackColor = ColorDialog1.Color

End If

End Sub

Private Sub closeButton\_Click() Handles closeButton.Click

' Close the form.

Close()

End Sub

Private Sub CheckBox1\_CheckedChanged() Handles CheckBox1.CheckedChanged

' If the user selects the Stretch check box, change

' the PictureBox's SizeMode property to "Stretch". If the user

' clears the check box, change it to "Normal".

If CheckBox1.Checked Then

PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage

Else

PictureBox1.SizeMode = PictureBoxSizeMode.Normal

End If

End Sub

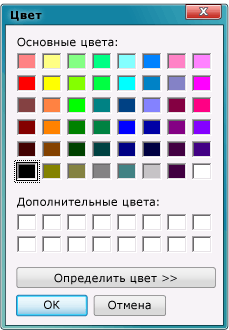
**Шаг 11. Запуск программы и изучение других функций**

.

Разработка программа завершена и она готова для выполнения. Можно запустить программу и задать цвет фона для PictureBox. В качестве дополнительного занятия, попробуйте улучшить программу с помощью изменения цвета формы, настройки кнопок и флажков, изменения свойств формы.

Запуск программы и настройка цвета фона

1. Нажмите F5 или в строке меню выберите **Отладка**, **Начать отладку**.
2. Прежде чем открыть изображение нажмите кнопку **Установить цвет фона**. Откроется диалоговое окно **Цвет**.

  
Диалоговое окно "Цвет"

1. Выберите цвет фона для элемента управления PictureBox. Внимательно просмотрите метод backgroundButton\_Click(), чтобы понять, как он работает.

|  |
| --- |
| **System_CAPS_ICON_note.jpg Примечание** |
| Можно загрузить рисунок из Интернета, путем вставки URL-адреса в диалоговое окно **Открыть файл**. Попробуйте найти изображение с прозрачным фоном, таким образом, чтобы был показан цвет фона. |

1. Нажмите кнопку **Очистить рисунок**, чтобы обеспечить удаление рисунка. Затем выйдите из программы, нажав кнопку **Закрыть**.

**Изучение других функций**

* Измените цвет формы и кнопок с помощью свойства **BackColor**.
* Настройте кнопки и флажки с помощью свойств **Font** и **ForeColor**.
* Измените свойства формы **FormBorderStyle** и **ControlBox**.
* Используйте свойства **AcceptButton** и **CancelButton**, чтобы кнопки автоматически нажимались, когда пользователь нажимает клавиши ВВОД или клавиши ESC. Сделайте так, чтобы программа открывала диалоговое окно **Открыть файл** при нажатии пользователем клавиши ВВОД и закрывала окно при нажатии клавиши ESC.

