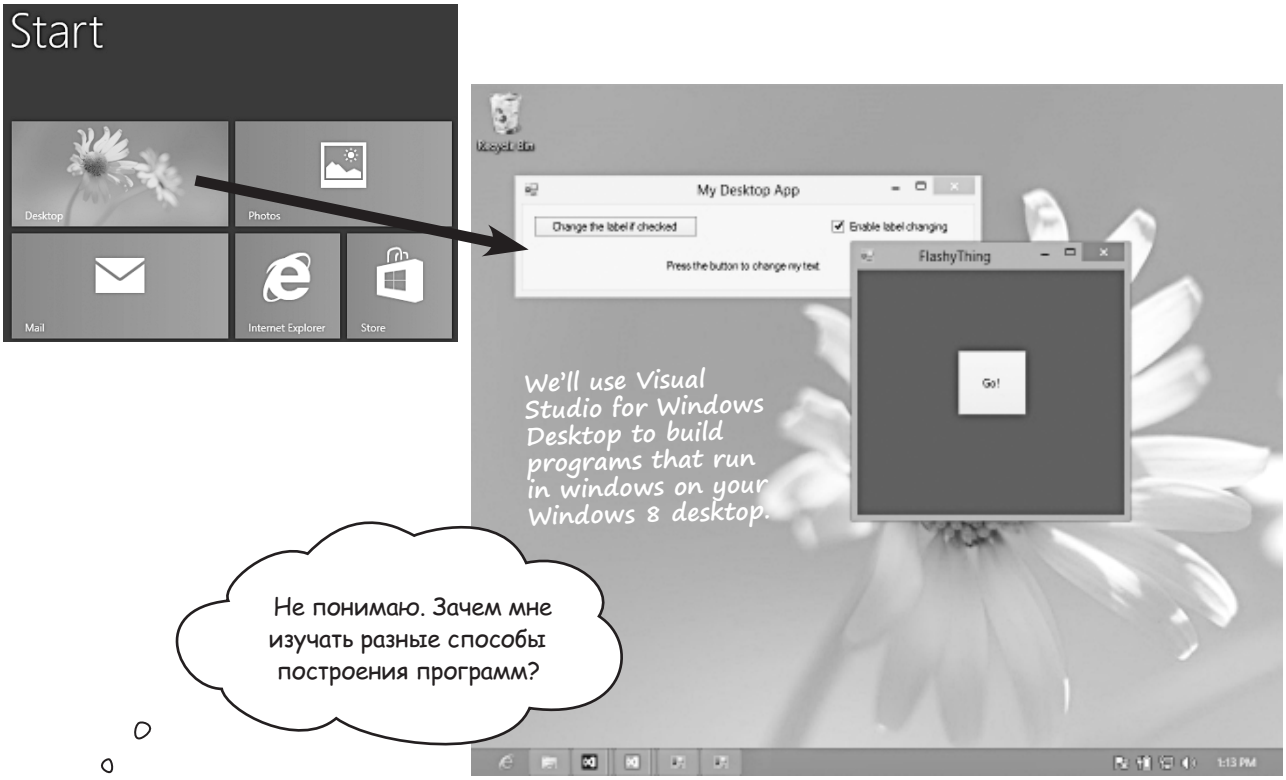
Приложения для рабочего стола Windows

Windows 8 познакомила нас с приложениями для магазина Windows и совершенно новым способом применения программного обеспечения. Но Visual Studio можно использовать и для построения **приложений для рабочего стола Windows** в виде окон на рабочем столе.



Приложения для рабочего стола Windows как обучающий инструмент

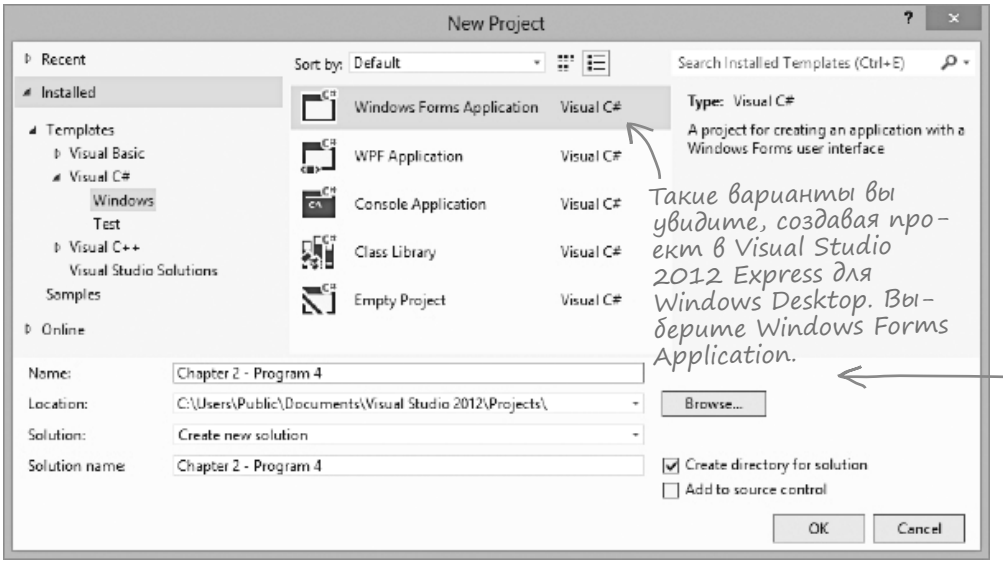
В нескольких следующих главах мы будем пользоваться Visual Studio для рабочего стола Windows и только потом вернемся к приложениям для магазина Windows. Дело в том, что первые во многом проще вторых. Они не настолько красивы, не интергируются с Windows 8 и не предоставляют такого удобного пользовательского интерфейса, как приложения для магазина Windows. Но для эффективного построения последних вам нужно познакомиться с рядом важных фундаментальных понятий. И проще всего начать, программируя для рабочего стола Windows. Сразу после этого мы вернемся к построению приложений для магазина Windows.

Изучая программирование для рабочего стола Windows, вы увидите альтернативные способы реализации многих вещей. Это позволяет лучше усвоить самое важное. Перевернite страницу, чтобы понять, что мы имеем в виду...

Перестроим приложение для рабочего стола Windows

Упражнение!

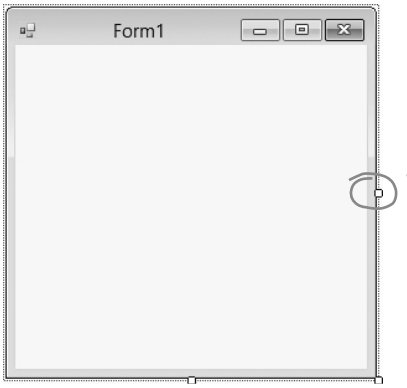
Запустите Visual Studio 2012 для рабочего стола Windows и создайте новый проект. На этот раз вам предлагается другое меню. Раскройте разделы Visual C# и Windows и **выберите Windows Forms Application**.



Такие варианты вы увидите, создавая проект в Visual Studio 2012 Express для Windows Desktop. Выберите Windows Forms Application.

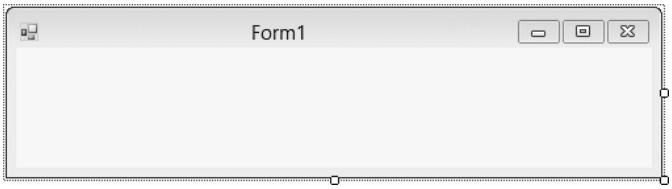
Имеет смысл выбирать более удобные названия, чем «Chapter 2 — Program 4», но в данном случае специально использовано имя с пробелами и дефисом, чтобы показать, как это повлияет на пространство имен.

1 **Приложения Windows Forms начинаются с формы с редактируемыми размерами.** Приложение Windows Forms в конструкторе представляется в виде окна. Сделаем его размером 500×130. Найдите на границе формы маркер и перетащите его. Обратите внимание, как при этом меняются цифры в строке состояния IDE. Двигайте его, пока не увидите в строке состояния значение 500 x 130 .

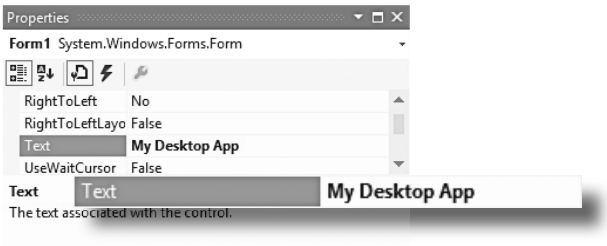


Двигайте такие маркеры, пока форма не приобретет нужный размер.

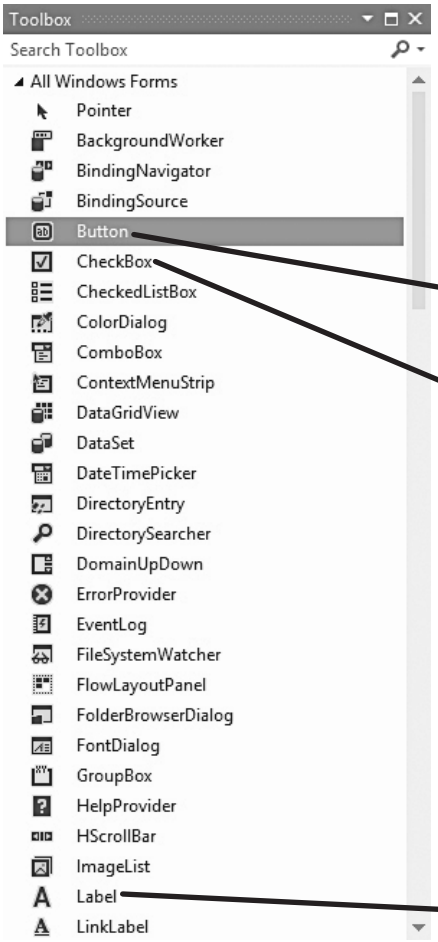
Вот какой вид должна приобрести наша форма.



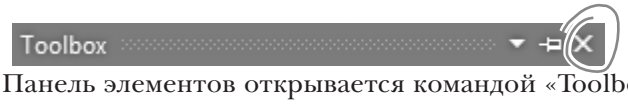
2 **Меняем заголовок формы.**
По умолчанию в заголовке формы написано «Form1». Выделите форму щелчком и поменяйте свойство `Text` в окне Properties.



3 **Добавим кнопку, флажок и метку.**
Перетащите с панели элементов на форму элементы Button, CheckBox и Label.



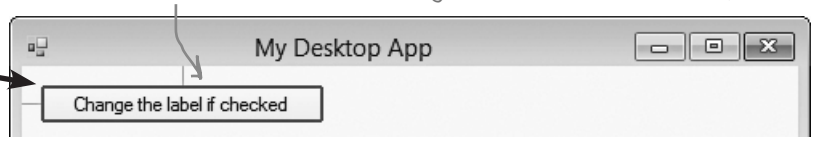
Подсказка: для придания элементу Label нужного рамера используйте AutoSize.



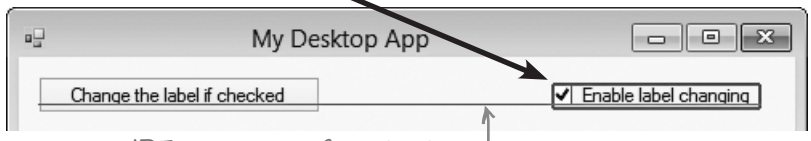
Toolbox

Панель элементов открывается командой «Toolbox» из меню View или щелчком на вкладке Toolbox. Ее можно закрепить, щелкнув на кнопке в виде булавки (📌) в правом верхнем углу окна Toolbox, или перетащить в сторону, превратив в плавающее окно.

Разделительные линии помогают указать положение элемента.



На следующей странице мы займемся окном Properties для изменения текста и задания состояния элемента CheckBox. Попробуйте самостоятельно понять, как это делается!



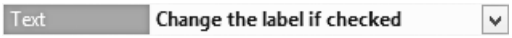
IDE помогает выравнивать элементы управления, отображая в процессе их перетаскивания направляющие линии.

Корректный выбор Visual Studio

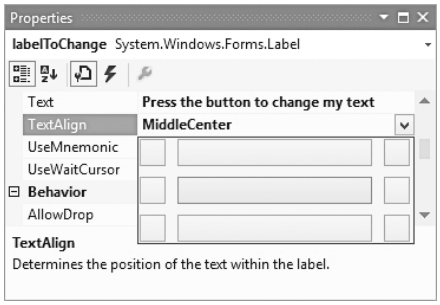
*Пользователям версии Express Visual Studio 2012 нужно установить два варианта приложения. Мы пользовались Visual Studio 2012 для Windows 8 при создании приложений для магазина Windows. А теперь нам нужен **Visual Studio 2012 для Windows Desktop**. К счастью, обе версии Express можно бесплатно скачать на сайте Microsoft.*

4 Настраиваем элементы управления в окне Properties.

Щелчком выделите элемент Button и в окне Properties задайте свойство `Text`:



Измените свойство `Text` элементов CheckBox и Label в соответствии со снимком экрана на следующей странице и присвойте свойству `Checked` элемента CheckBox значение `True`. Затем выделите элемент Label и присвойте свойству TextAlign значение `MiddleCenter`. Кроме того, **дайте элементам имена**: кнопке — Button `changeText`, флажку — CheckBox `enableCheckbox`, а метке — Label `labelToChange`. Внимательно посмотрите на приведенный ниже код и определите, каким образом эти имена там используются.



Свойство `AutoSize` элемента Label поменяйте на `False`. По умолчанию метки подгоняют свой размер под текст. Убрав у параметра `AutoSize` значение `true`, мы сделаем видимыми манипуляторы, чтобы **растянуть метку на ширину окна**.

5 Добавим к кнопке метод обработчика событий.

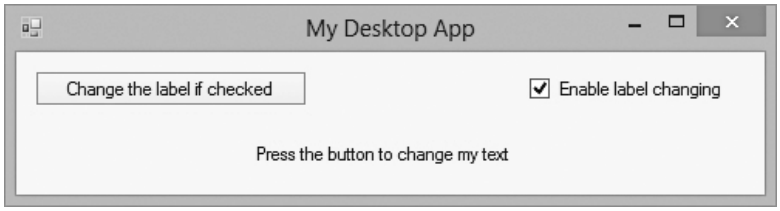
Дважды щелкните на кнопке, чтобы добавить метод обработчика событий. Вот его код:

```
Form1.cs
Chapter_2___Program_4.Form1
Form1()
namespace Chapter_2___Program_4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void changeText_Click(object sender, EventArgs e)
        {
            if (enableCheckbox.Checked == true)
            {
                if (labelToChange.Text == "Right")
                {
                    labelToChange.Text = "Left";
                    labelToChange.TextAlign = ContentAlignment.MiddleLeft;
                }
                else
                {
                    labelToChange.Text = "Right";
                    labelToChange.TextAlign = ContentAlignment.MiddleRight;
                }
            }
            else
            {
                labelToChange.Text = "Text changing is disabled";
                labelToChange.TextAlign = ContentAlignment.MiddleCenter;
            }
        }
    }
}
121 %
```

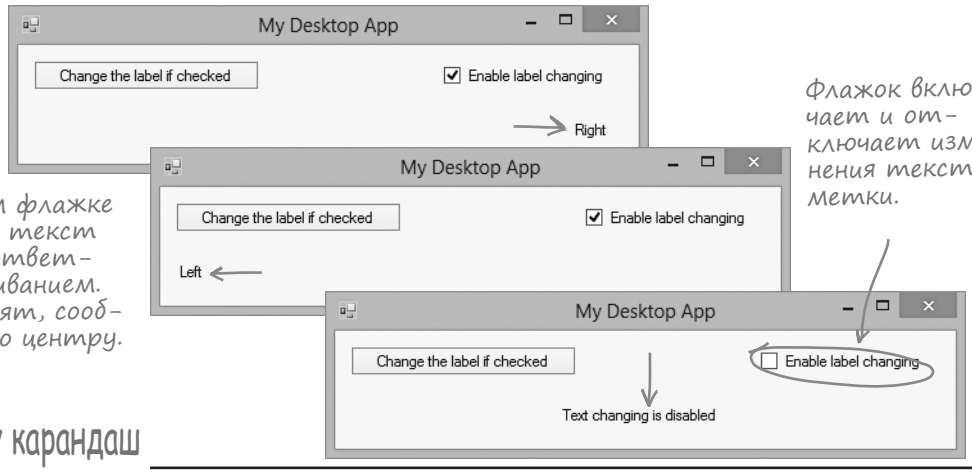
При двойном щелчке на кнопке IDE сгенерировала обработчик события и присвоила ему имя changeText_Click(), в соответствии с именем кнопки changeText.

Это код для метода обработчика события. Внимательно его прочитайте. Можете найти его отличия от аналогичного кода в предыдущем упражнении?



Отладим программу в IDE.

Затем IDE построит программу и запустит ее, открыв окно. Попробуйте щелкать на кнопке и менять состояние флажка.



При установленном флажке метка показывает текст Left или Right с соответствующим выравниванием. Если же флажок снят, сообщение выводится по центру.

Флажок включает и отключает изменения текста метки.

Возьми в руку карандаш

Опишите назначение различных строк кода, как показано в примере.

```
using System;
using System.Linq;
using System.Text;
using System.Windows.Forms;


namespace SomeNamespace
{
        class MyClass {

                public static void DoSomething() {

                        MessageBox.Show("Здесь будет сообщение");

                }

        }
}
```

Оператор "using" добавляет в классы C# методы из других пространств имен

Вы еще не встречались с функцией MessageBox, но она часто используется в приложениях для рабочего стола. И подобно большинству классов и методов, она носит значимое имя.

Решение на странице 131

Начало работы программы

При создании нового приложения Windows Forms IDE добавляет файл Program.cs. Дважды щелкните на его имени в окне Solution Explorer. Файл содержит класс Program, обладающий методом Main(). Этот метод представляет собой **точку входа**, то есть именно отсюда программа начинает свою работу.

Приложения для рабочего стола выглядят по-другому, и это хорошо для обучения.

Приложения для рабочего стола Windows выглядят примитивнее приложений для магазина Windows, потому что для них сложнее создать усовершенствованный интерфейс пользователя. Но на данном этапе это хорошо, так как ничто не будет отвлекать вас от изучения ключевых понятий C#. А когда мы вернемся к приложениям для магазина Windows, вам будет проще.

Этот код, автоматически созданный в упражнении из предыдущей главы, вы найдете в файле Program.cs.

Код под увеличительным стеклом

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Chapter_2___Program_4
{
    static class Program
    {
        /// <summary>
        /// Точка входа в приложение.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

1. *Имя этого пространства имен IDE сгенерировала, взяв за основу имя проекта. В данном случае проект назывался "Chapter 2 - Program 4". Имя с пробелами и дефисом было выбрано намеренно, чтобы продемонстрировать, как IDE превращает их в нижние подчеркивания.*
2. *Начинающиеся со слэшей строки являются комментариями, и их можно добавлять куда угодно. Компилятор их просто не видит.*
3. *Программа начинает работу с точка входа.*
4. *Этот оператор создает и отображает форму Contacts, а также завершает программу при закрытии формы.*

Первая часть имени класса или метода называется объявлением.

На стадии начального знакомства с кодом вам требуется понять, на что следует обращать внимание.

Существует несколько тонких моментов в разработке приложений. Вы еще развлечетесь с ними на следующих страницах. Но большую часть вашей работы будет составлять перетаскивание элементов управления и редактирования кода C#.

1. **Встроенные функции C# и .NET.**
Подобные строки находятся в верхней части почти всех файлов классов C#. `System.Windows.Forms` — это **пространство имен**. Строка `using System.Windows.Forms` дает программе доступ ко всем объектам этого пространства, в данном случае к визуальным элементам — кнопкам и формам.

Постепенно ваши программы будут содержать все больше пространств имен.

Без строчки using вам придется в явном виде вводить System.Windows. Forms при обращении к объекту из этого пространства имен.

2. **Выбор пространства имен для кода.**
IDE называет созданное пространство имен в соответствии с именем проекта. Именно к этому пространству относится весь код.

Пространства имен позволяют использовать одни и те же имена в различных программах, при условии, что программы не принадлежат к одному пространству.

3. **Код принадлежит к конкретному классу.**
В вашей программе этот класс называется `Program`. Он содержит код запуска программы и код вызова формы Form 1.

В одном пространстве имен может находится несколько классов.

4. **Наш код содержит один метод, состоящий из нескольких операторов.**
Внутри любого метода может находиться произвольное количество операторов. В нашей программе именно операторы вызывают форму.

Технически программа может иметь несколько методов Main(), нужно только указать, какой из них будет точкой входа.

5. **Точка входа.**
Каждая программа на C# **должна** иметь только один метод с названием `Main`. Именно он выполняется первым. C# проверяет классы на его наличие, пока не находит строчку `static void Main()`. После этого выполняется первый и все следующие за ним операторы.

Любое приложение на C# должно иметь единственный метод Main. Он является точкой входа для вашего кода.

При запуске кода метод Main( ) выполняется ПЕРВЫМ.

Редактирование точки входа

В программе главное — точка входа. При этом не имеет значения, к какому классу принадлежит содержащий ее метод и какие действия производит. Нет ничего таинственного в том, как это работает. Вы можеет проверить это самостоятельно, изменив точку входа.

1. Вернитесь к программе, которую мы только что написали. В файле `Program.cs` присвойте методу `Main` имя `NotMain` и **попробуйте построить и запустить** программу. Что произойдет?

2. Создадим новую точку входа. **Добавьте класс** с именем `AnotherClass.cs`. Для этого щелкните правой кнопкой мыши на имени файла в окне Solution Explorer и выберите команду Add>>Class... IDE добавит в программу класс `AnotherClass.cs`. После этого код примет вид:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Chapter_2___Program_4
{
    class AnotherClass
    {
    }
}
```

В файл были добавлены четыре стандартные строчки с оператором `using`.

Этот класс находится в том же пространстве имен.

Имя присваивается классу автоматически (на основе имени файла).

3. Добавьте в верхнюю часть строку **`using System.Windows.Forms;`**
Не забудьте, что в конце строки должна стоять точка с запятой!

4. Добавьте этот метод к классу **`AnotherClass`**, написав его внутри фигурных скобок:

*Класс **MessageBox** принадлежит пространству имен System.Windows.Forms, поэтому на шаге #3 вы и добавили оператор using. Метод **Show ()** является частью класса MessageBox.*

```
class AnotherClass
{
    public static void Main()
    {
        MessageBox.Show("Pow!");
    }
}
```

В C# регистр букв имеет значение! Обращайте внимание на прописные и строчные буквы.