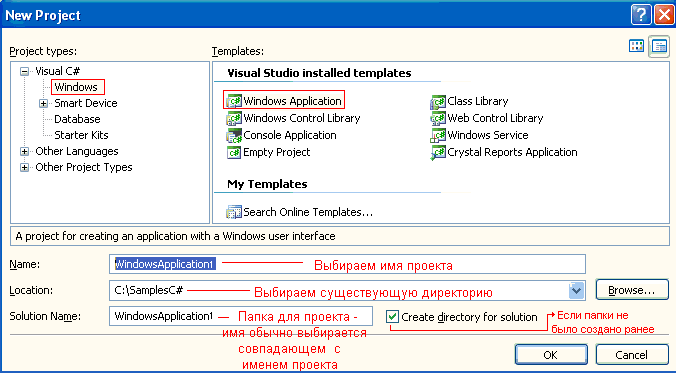
**Laboratory No. 1. Project creation. Working with forms. Buttons**

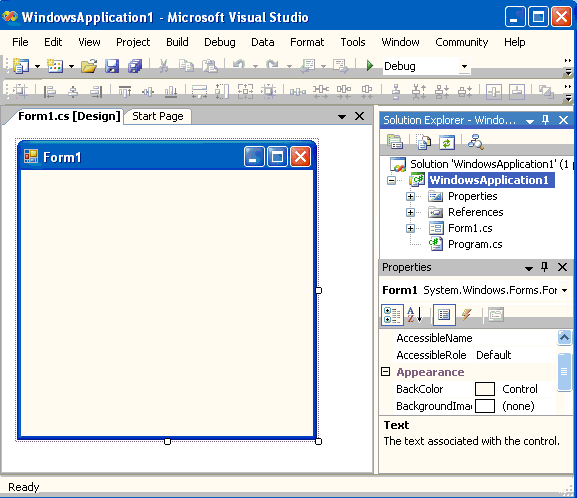
To create a Windows Application C # solution (application, project), after starting Visual Studio.NET, click on the button in the New Project (left on the toolbar), or press the key combination Ctrl + Shift + N, or go to the File menu and then select New and then Project. The New Project dialog box appears on the screen (Fig. 1.).

Fig. 1. Dialog panel New Project

Select and set:

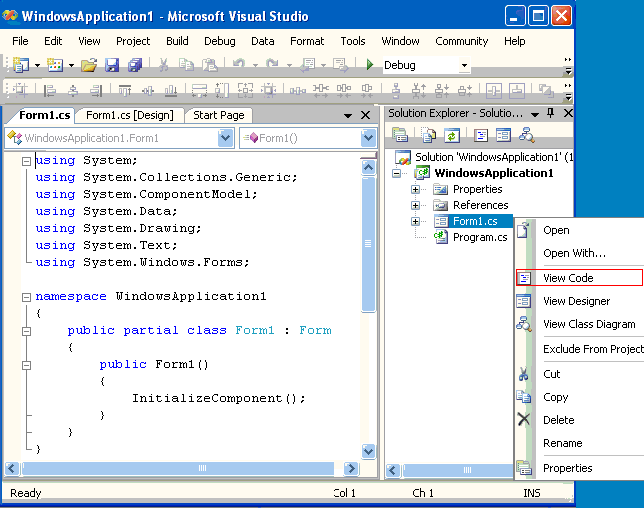
* Project Types: Visual C #;
* Templates: Windows Application;
* Name: write the name with which the .exe file will be generated;
* Location: directory in which the project will be placed (for convenience, it is desirable to create a directory closer to the root of the disk, for example C: \ SamplesC #);
* You can check the "Create Directory for Solytion" checkbox or set the directory for this solution in the selected folder (Browse ...).
* Click OK.

A blank window appeared in the code editor on the Form1.cs [Designe] tab (Fig. 2.1.), And in the C: \ SamplesC # folder the WindowsApplication1 folder with the WindowsApplication.sln file (solution project file) and the WindowsApplication subfolder with its Bin, Obj subfolders , Properties, which so far contain empty subdirectories and, into which the files necessary for building and debugging the program and the .exe file will be placed, which will be discussed below. The project files Form1.cs, Form1.Designer.cs, Program.cs, and WindowsApplication.csproj are located in the WindowsApplication1 subfolder.

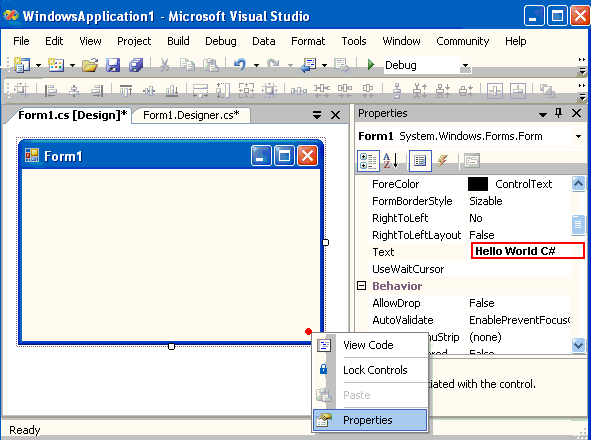


Rice. 2.1.

In the "Solutation Explorer" window (Menu Viev | Solutation Explorer), right-click on the project file - "Form1.cs" - and select the "Viev Code" menu item (or press F7), call the project code file (Fig. 2.2. ):

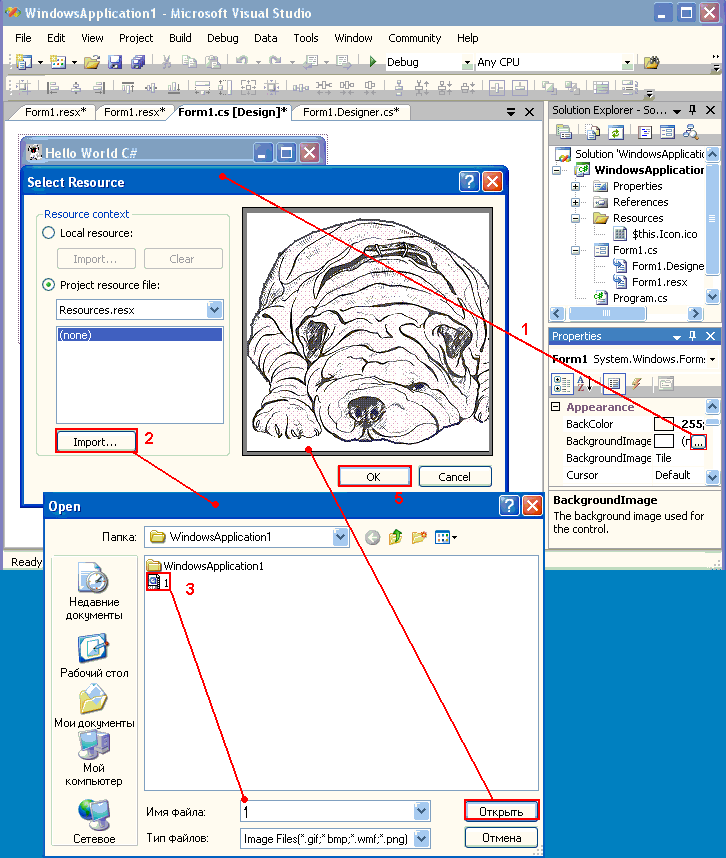
Fig. 2.2.

Return to the form designer (Form1.cs [Design] tab), call the form properties (Properties) in the form's context menu (Fig.2.5.) By right-clicking on the form's form (or View \ Properties Window menu Ctrl + W, P ) and in the Properties window from the set of properties set:



* Property Text - Hello World C #;
* BackColor - select (click on the field and press the down arrow) the shape color we like out of three possible sets;
* StartPosition - CenterScreen - from what position the window will be located after launch.

And the final touch - in order to make a C # business card, instead of the colors in the BackGround, add a picture to the form's BackgroundImage property, as shown in Figure 3.2., And, do not forget to set the BackgroundImageLayout property to Stretch.



Rice. 3.2.

Now press F5 (Debug / Start Debuging menu) and see the created application:

**Buttons**

In the View menu | Toolbox, we find the Button control and, by clicking on it with the mouse, move the pointer to the form (if we made a mistake, then click on the desired control or outside the form window) - and then, click with the mouse within the borders of the form window - the control has moved to the form.

Next, we set the size of the button. The sizes of any control, as well as the application as a whole, can be set more clearly by dragging the markers and dragging the entire control (with the control selected by the mouse click and the left mouse button pressed).

The next stage is setting the properties of the button, which are set in the same way as setting properties for the form itself (View | Proporties Window or the context menu of the control, item Properties):

Set the following properties:

* Text - the name of the button;
* Font - font size and type;
* BackColor - color;
* FlatStyle - appearance.

Go to the Events tab of the Properties window and double-click the left mouse button in the field opposite the Click event (for a click event, this is equivalent to a double mouse click on the button itself). After that, the handler code will appear in the Form1.cs file of the solution:

private void

button1\_Click (object sender, EventArgs e)

{

}

Enter the following code into it to display the dialog box:

private void button1\_Click (object sender, EventArgs e)

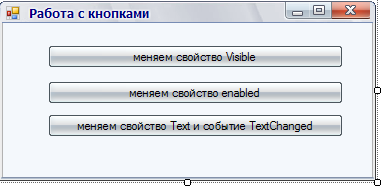
{

MessageBox.Show ( this, "Hello World C #");

}

**Basic properties and events of the Button component**

Create a form



On the button, change the Visible property, create a Click:event

private void button1\_Click (object sender, EventArgs e)

{

button1.Visible = false;

}

This property allows you to hide the component, but this component remains on the form, this property is available for all components.

Change the enabled property to the Button and create a Click:event

private void button2\_Click (object sender, EventArgs e)

{

button2.Enabled = false;

}

Enabled - makes components unavailable, but leaves them visible on the form. All components have this property.

The values ​​of the Enabled and Visible properties can be checked in the conditional if statement, let's change the previous code:

private void button2\_Click (object sender, EventArgs e)

{

button2.Enabled = false;

if (button1.Visible == false) MessageBox.Show ("The first button is not visible");

}

Change the Text property to the button and create the TextChanged event Click:

private void button3\_Click (object sender, EventArgs e)

{

button3.Text = "Clicked on the button";

}

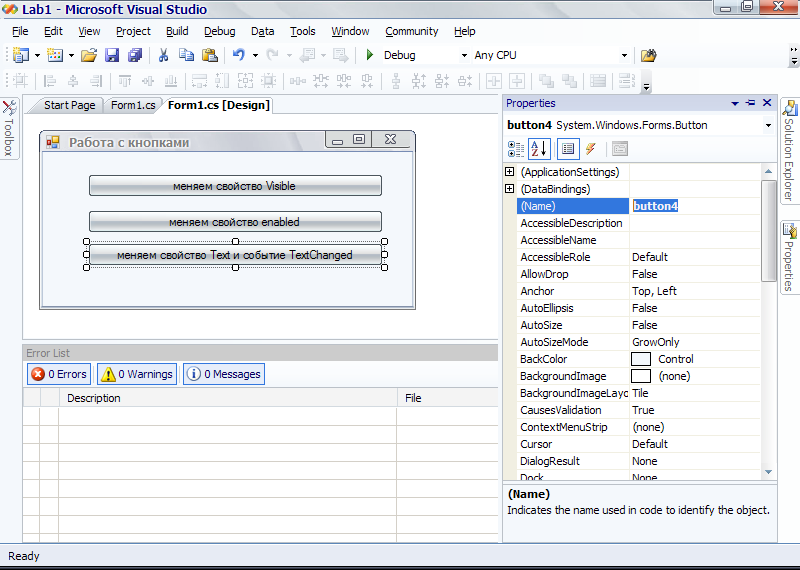
And the TextChangedevent

private void button3\_TextChanged (object sender, EventArgs e)

{

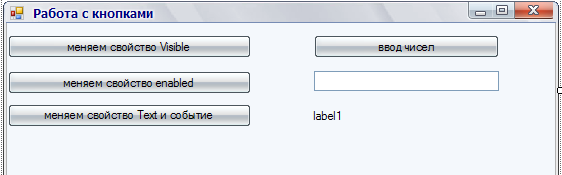
MessageBox.Show ("An event has occurred");

}

Errors: Take a close look at the Name property of the components. 

Given the given name, the previous code will cause an error or incorrect work of the project, since you are referring to button3.Text = "Clicked on the button";

Add to our form



On the button for entering numbers, create an event Clik

private void button3\_Click (object sender, EventArgs e)

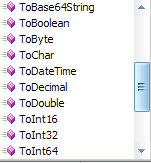
{

int viNumber = Convert.ToInt32 (textBox1.Text) \* 5;

label1.Text = "Entered number multiplied by 5 =" +

viNumber.ToString ();

}

Functions Convert.ToInt32 (textBox1.Text) is needed to convert a value from a string to an integer, there are many variations ofdepending on the task Convert. (textBox1.Text)

label1.Text = "Entered number multiplied by 5 =" +

viNumber.ToString ();

This line changes the Text property. Let's stop at viNumber.ToString (); since the Text property of the display components has a string type, you need to call the function to convert to a string ToString ();

Exercise 1. Create a form



By clicking one of the buttons, the value is displayed in the TextBox

Exercise 2. Using the previous project, change to the following, the operation sign is entered from the keyboard, use the if constructs to determine the operation sign

