

# Visual C# Program: Temperature Conversion Program

---

In this chapter, you will learn how to use the following Visual C# Application functions to World Class standards:

- **Writing a Temperature Conversion Program**
- **Opening Visual C# Editor**
- **Beginning a New Visual C# Project**
- **Laying Out a User Input Form in Visual C#**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Insert a Label into a Form to Post an Output**
- **Insert a GroupBox and Two Radial Buttons into a Form**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual C# to Communicate the Copyright**
- **Declaring Variables in a Program**
- **Setting Variables in a Program**
- **Using a Condition Statement and Label to Communicate the Answer**
- **Resetting the Data**
- **Exiting the Program**
- **Running the Program**

## Writing a Temperature Conversion Program

---

Many times, we will need a simple application just to convert numbers from one unit of measure to another such as Fahrenheit to Celsius and from Celsius to Fahrenheit. This program will do just that function. In order to accomplish the project, we need to know the formula to accomplish the translation.

The equation to convert Celsius to Fahrenheit is:

$$F = \frac{9}{5}C + 32$$

And the equation to convert Fahrenheit to Celsius is:

$$C = \frac{5}{9}(F - 32)$$

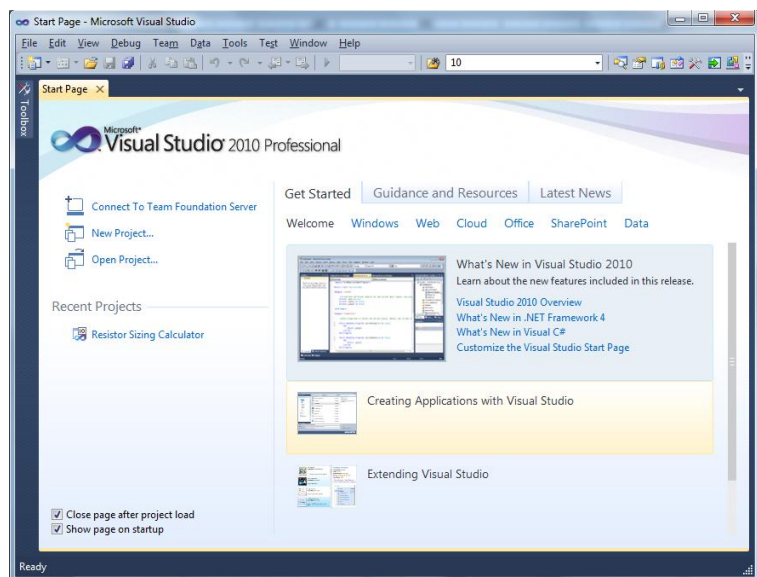
Although this is simple code to write, many larger programs are just a collection of smaller and efficient subroutines, so writing smaller applications is just a stepping stone to larger projects.

## Open the Visual C# Editor

---

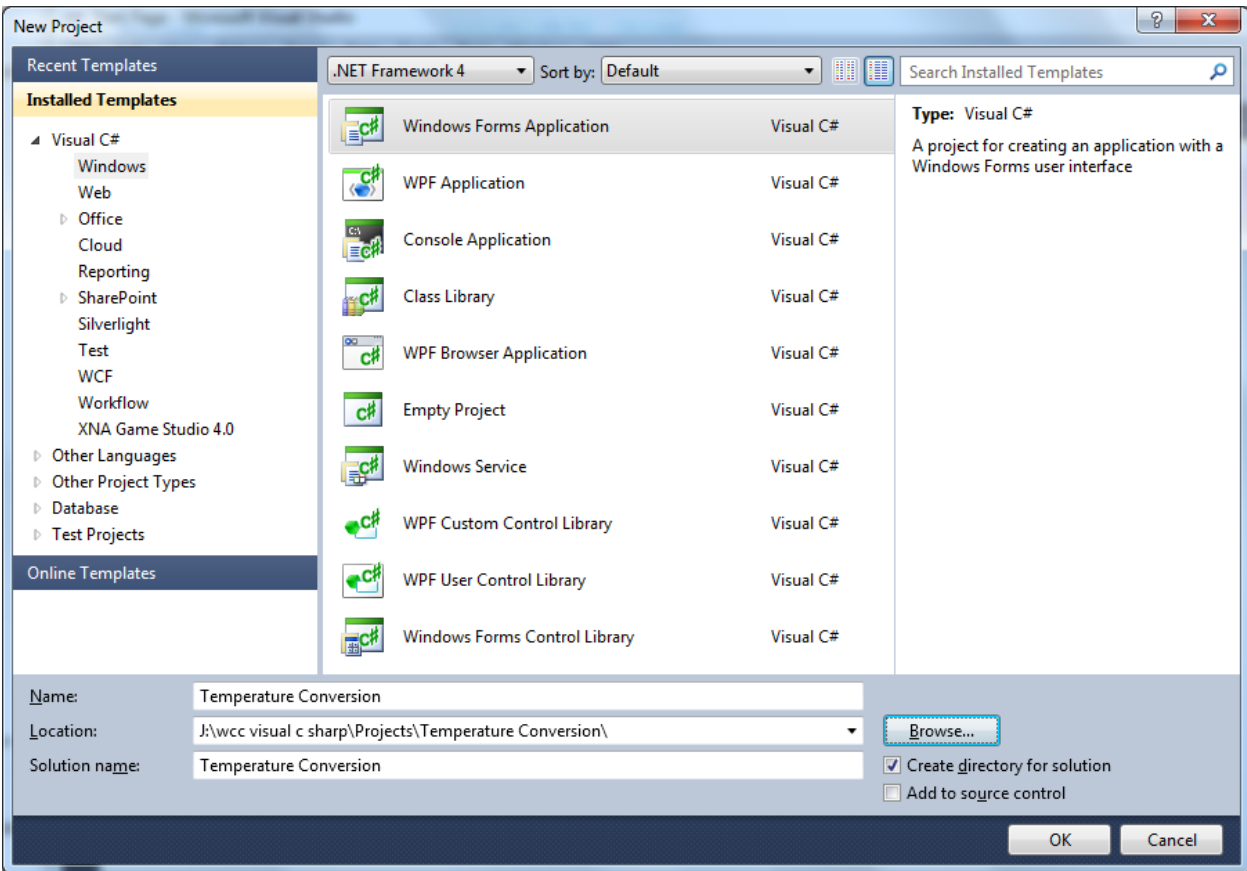
In this lesson, we will step through each procedure in adding labels, textboxes, group boxes, radial and command buttons and we will integrate into the tutorial the methods to add, subtract, multiply and divide numbers. We will use an If then else condition statement to create the answer in a label. As in every project, we will create a variable, set the value, execute mathematical equations and output data.

To open a new project, we will select New Project on the left side of the Visual Studio window.



**Figure 4B.1 – The Start Page**

We start a new Windows Application by picking the Windows Application icon from the installed templates list on the New Project window.



**Figure 4B.2 – New Project**

We start a new Windows Application Project by picking the Windows under Visual C # in the left pan of the New Project window. Then we pick Windows Form Application in the center pane.

At the bottom of the Window, we name the project, Temperature Conversion. We make a folder for our projects called Visual C Sharp on the desktop, on our flash drive or in the Documents folder. We make another folder inside the first called Temperature Conversion. On the New Project window, we browse to the Temperature Conversion location. The solution name is the same as the project name.

## Beginning a New Visual C# Application

Remember, that all programming projects begin with one or more sketches. The sketch will show labels, textboxes, a group box, radial and command buttons. In this project, we will name the input form, **Temperature Conversion**. We will have one textbox to key in the temperature. We will have two labels, one with a border and one without a border to output the converted temperature and the unit of measure. We will have one GroupBox that will hold two radial buttons to identify the temperature that we are converting. We will have three command buttons, **Convert**, **Reset** and **Exit**. On the bottom of the form, we will write the copyright

statement using another label. On this presentation, we can help ourselves by being as accurate as possible, by displaying sizes, fonts, colors and any other specific details which will enable us to quickly create the form. On this form, we will use a 12 point Arial font. From the beginning of inserting the form into the project, we need to refer to our sketch.

We should train new programmers initially in the art of form building. When using the editor, we insert and size the form, and selecting the Controls Toolbox, we will place all the various input tools and properly label them. Whenever we place an input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort to arrange the tool by performing such actions as naming, labeling and sizing the visual input device.

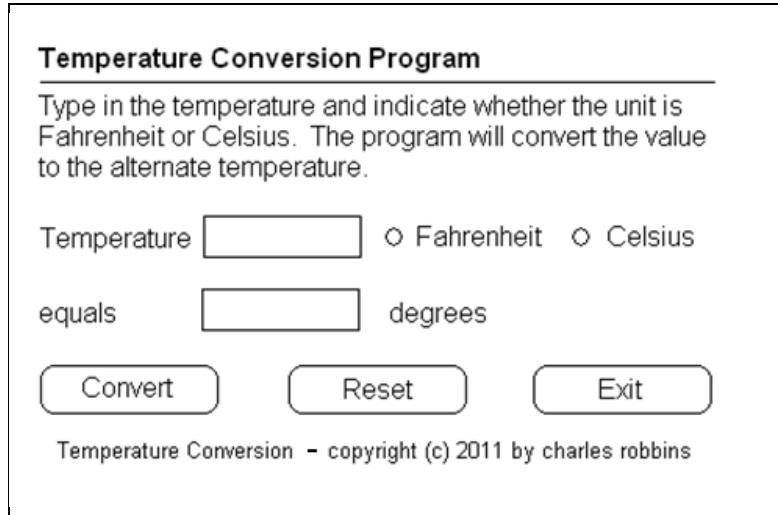


Figure 4B.3 – Sketch of the Resistor Sizing Form

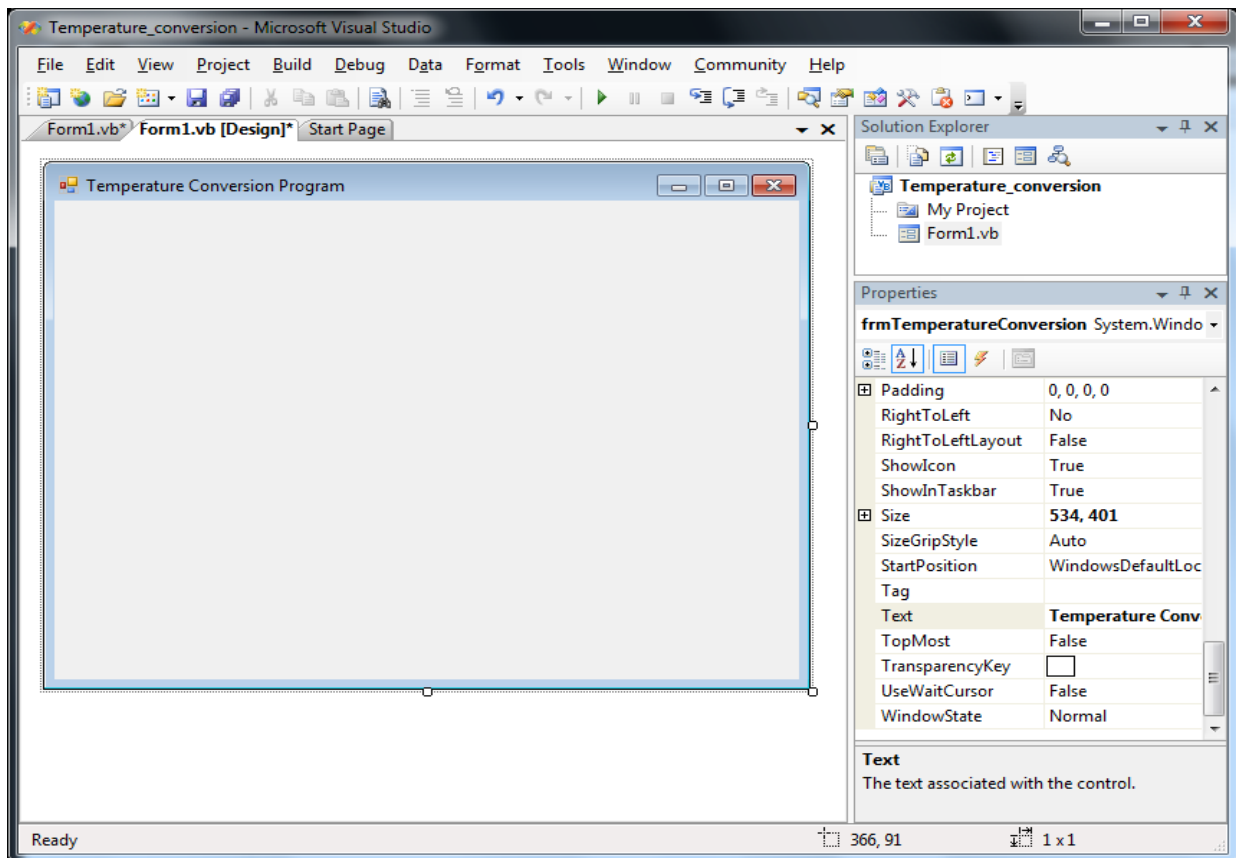


Figure 4B.4 – Designing the Temperature Conversion Form in Visual C#

## Laying Out a User Input Form in Visual C#

We will change the **Text** in the Properties pane to Temperature Conversion to agree with the sketch in Figure 4B.3. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
BackColor	LightSteelBlue
Font	Arial, 12 pt
Size	432,302

The first number is the width and the second number is the height. The form will change in shape to the size measurement.

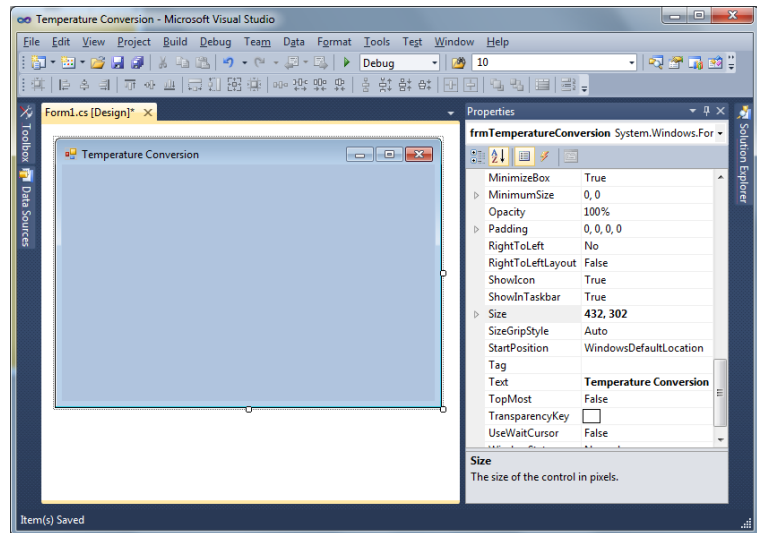


Figure 4B.5 – Setting the Form Properties

The background color will change to a LightSteelBlue. There are many more attributes in the Properties pane that we will use on future projects.

In this project, we will select the font in the form. By selecting the font, font style and size for the form, each label, textbox and command button we insert will have these settings for their font.

When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual C# Font window.

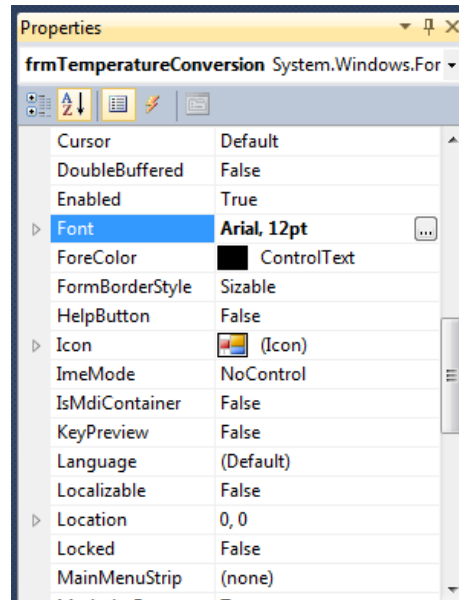


Figure 4B.6 – The Font Window in Visual C#

We will select the Arial font, Regular font style and 12 size for this project to agree with the initial sketch if the user input form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.

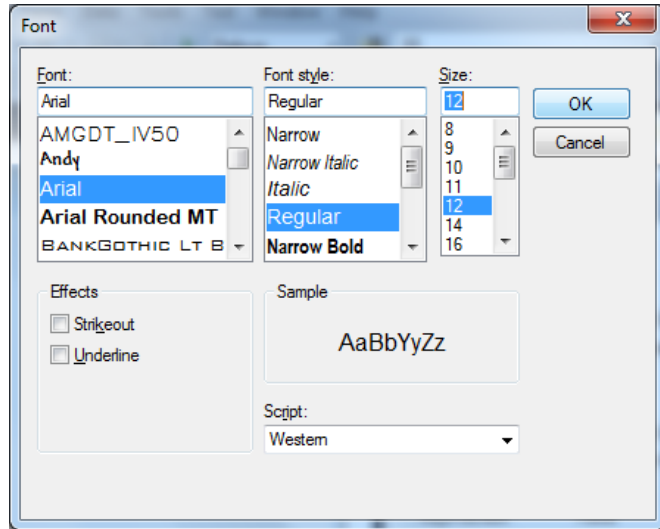


Figure 4B.7 – Changing the Font to Arial

## Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in Windows; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

We will insert our first Label on the upper left corner of the form and call the entity **lblInstruction**.

Alphabetic	
(Name)	lblInstruction
AutoSize	False
Text	Type in the temperature and indicate whether the unit is Fahrenheit or Celsius. The program will convert the value to the alternate temperature.

Since the bgcolor and font are already set.

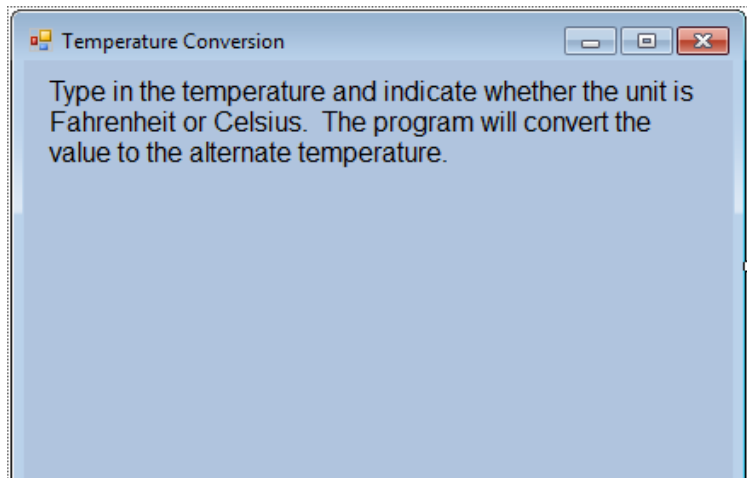


Figure 4B.8 – The Finished Label on the Form

We will insert our next Label below the entity **lblInstruction** as shown and call it **lblTemp**.

Alphabetic	
(Name)	lblTemp
Text	Temperature

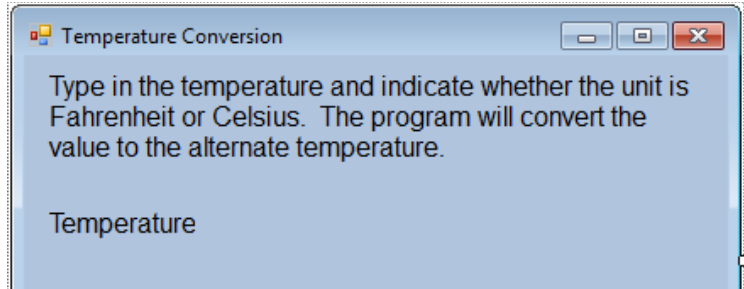


Figure 4B.9 – The Finished Temperature Label

## Inserting a Textbox into a Form

A textbox is used so that a user of the computer program can input data in the form of words, numbers or a mixture of both. Press the TextBox (ab) button on the Control Toolbar to add a textbox. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted textbox.

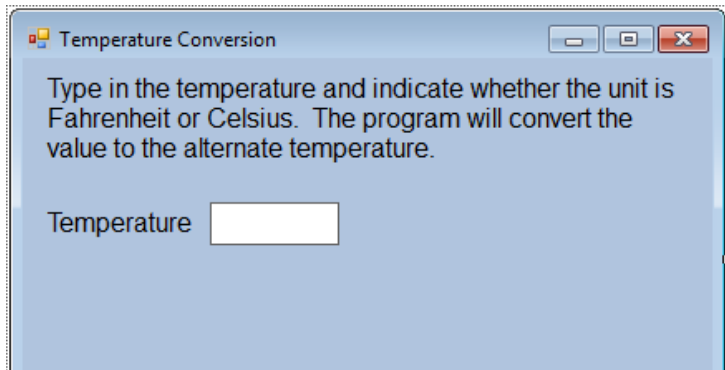


Figure 4B.10 – Placing a TextBox on the Form

We will name the TextBox using the three letter prefix followed by the name or phrase of the tool. For our first textbox, the name is **txtTemp**.

Alphabetic	
(Name)	txtTemp
BorderStyle	FixedSingle
Size	78,26
TextAlign	Center

The size of the textbox will be 78 wide and 26 tall and the characters inside the textbox will be center aligned. We will set the BorderStyle to FixedSingle to see a line.

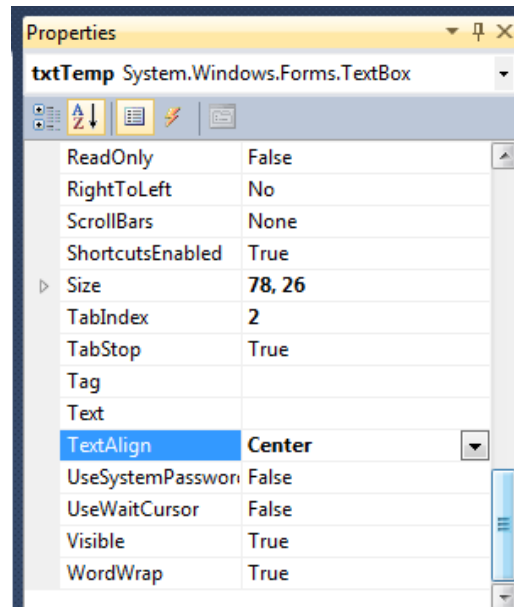


Figure 4B.11 – Setting the Size of the Textbox



We will insert our next Label on the upper left corner of the form and call the entity **lblEquals**.

Alphabetic	
(Name)	lblEquals
Text	equals

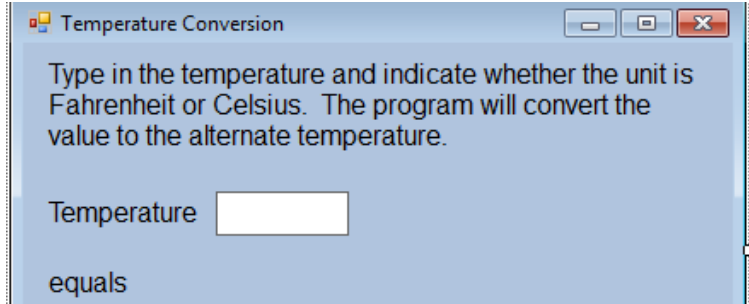


Figure 4B.12 – Adding another Label and Textbox

## Inserting a Label into a Form to Post the Output

Some labels on a form are in a position to display an answer after the user inputs data and they press the command button to execute the application. To add this label, press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

We will place a label under **lblTemp** label and call it **lblAnswer**. We will keep the label text clear. The key attributes for the label are:

Alphabetic	
(Name)	lblAnswer
Autosize	False
BackColor	White
BorderStyle	FixedSingle
Size	78,26
TextAlign	MiddleCenter

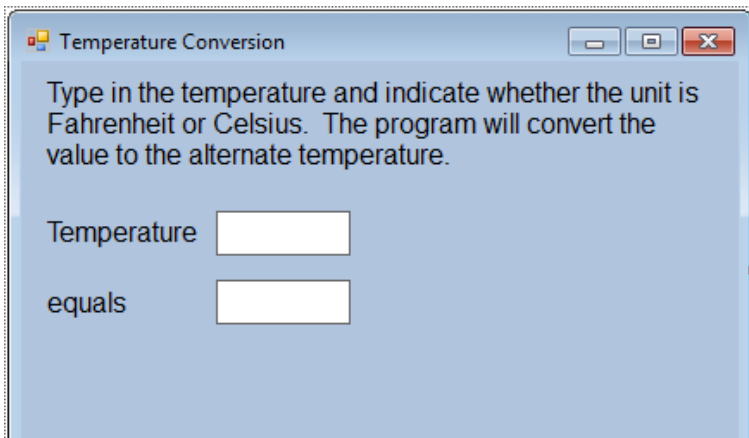


Figure 4B.13 – The Label for the Answer

We will insert the next label for the answer to the right of **lblAnswer** and name the label **lblUnitAnswer**.

Alphabetic	
(Name)	lblUnitAnswer
Autosize	True
TextAlign	MiddleCenter
Text	degrees

This label will contain the nit of measure for the answer.

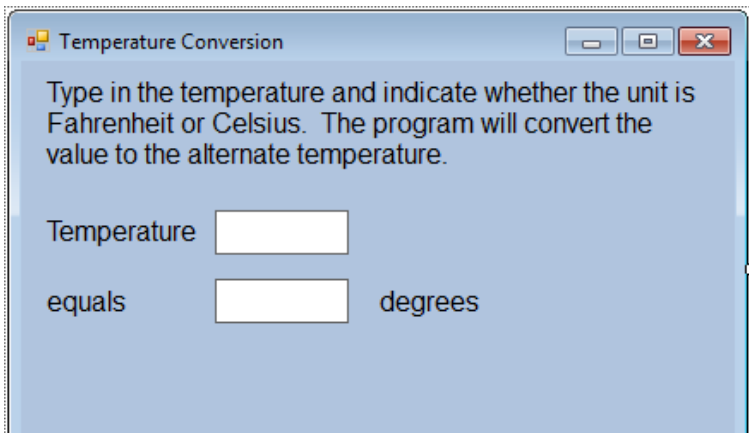


Figure 4B.14 – Another Label for the Answer



## Inserting a GroupBox and Two Radial Buttons onto the Form

When we want two radial or option buttons to toggle in an either or situation, then we will insert a group box into the form. To add the GroupBox, choose the entity from the Container list on the Toolbox. To size the GroupBox area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted Groupbox.

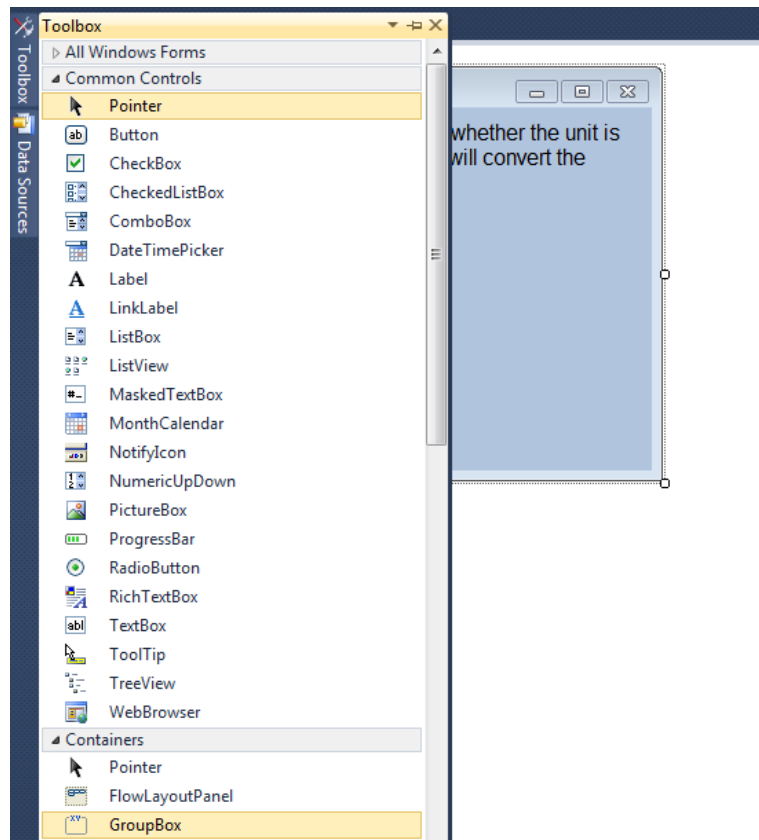


Figure 4B.15 – Selecting the GroupBox Container

To add the first option button to place inside the GroupBox, choose the Radial Button from the list on the Toolbox. To size the Radial Button area, click inside the GroupBox and make a rectangle. We will name the entity **radFahrenheit**.

Alphabetic	
(Name)	radFahrenheit
Text	Fahrenheit

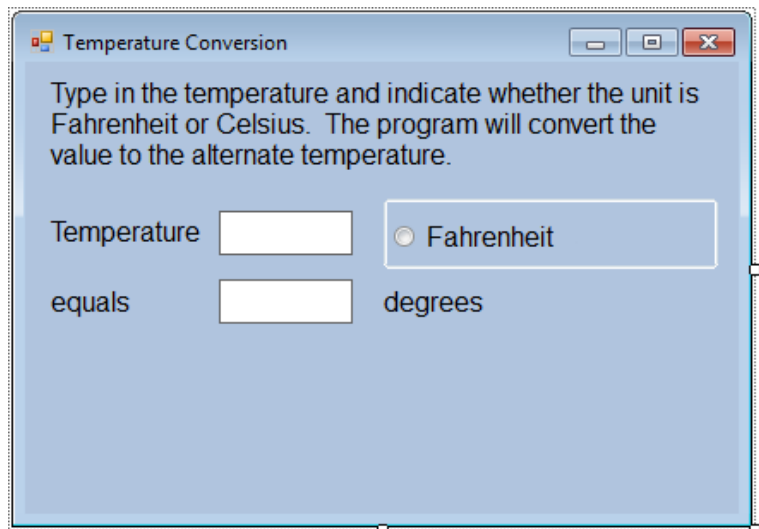


Figure 4B.16 – GroupBox and Radial Buttons

We will insert the next option button to the right of **radFahrenheit** and name the label **radCelsius**.

Alphabetic	
(Name)	radCelsius
Text	Celsius

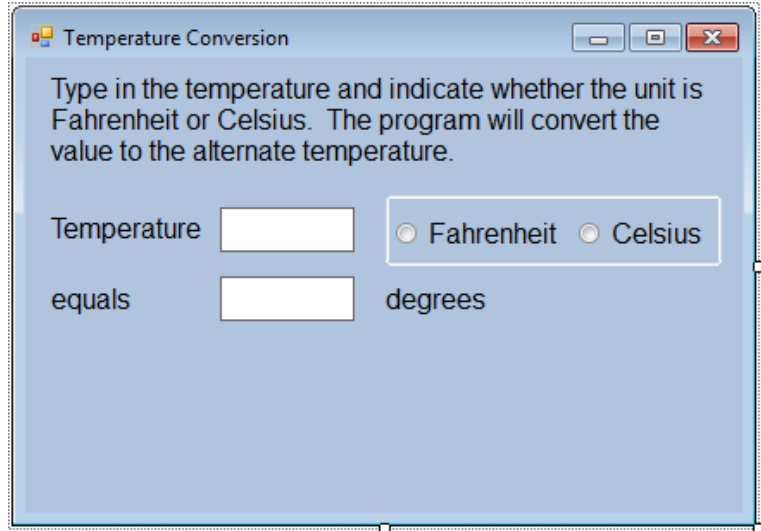


Figure 4B.17 – The Celsius Radial Buttons

## Inserting a Command Buttons into a Form

A command button is used so that a user will execute the application. Press the Command button on the Control Toolbar to add a command button. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the command button as shown in Figure 4B.18.

We will name the command button using the name is **cmdConvert**.

Alphabetic	
(Name)	cmdConvert
Caption	Convert
Size	99,30

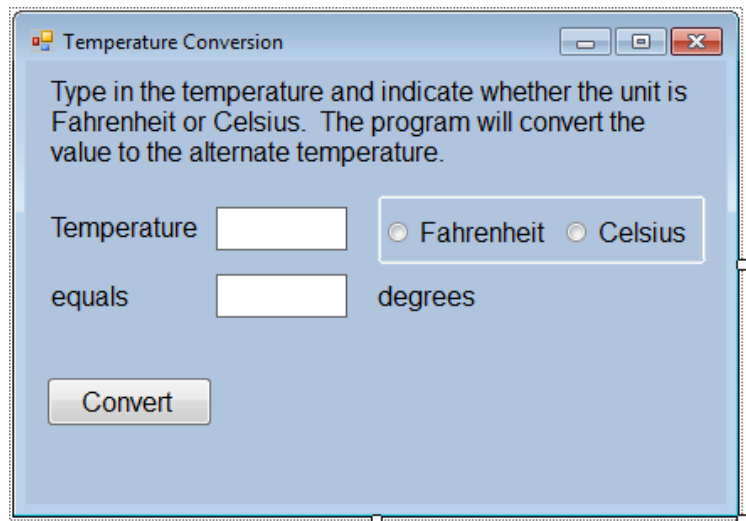


Figure 4B.18 – The Command cmdConvert Button

Add a second Command button, named cmdReset is for clearing the txtName and lblGreeting objects. The third command button is to exit the program. When the user presses the Exit command button, the application closes. Notice the equal spacing between the command buttons gives a visually friendly appearance.

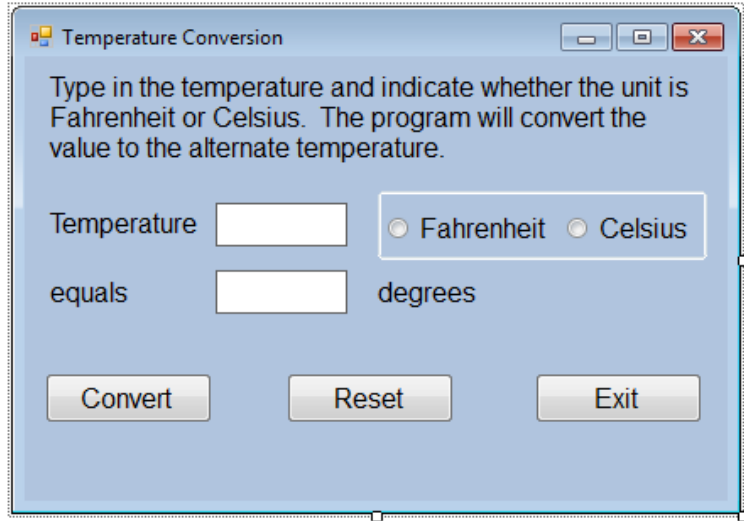


Figure 4B.19 – Insert Two More Command Buttons

## Adding a Copyright Statement to a Form

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

Finish the form with the following copyright information.

**Temperature Conversion -  
Copyright (c) 2011 by charles  
robbins**

If there are special rules or instructions that the user needs to know, place that information on the bottom of the form.

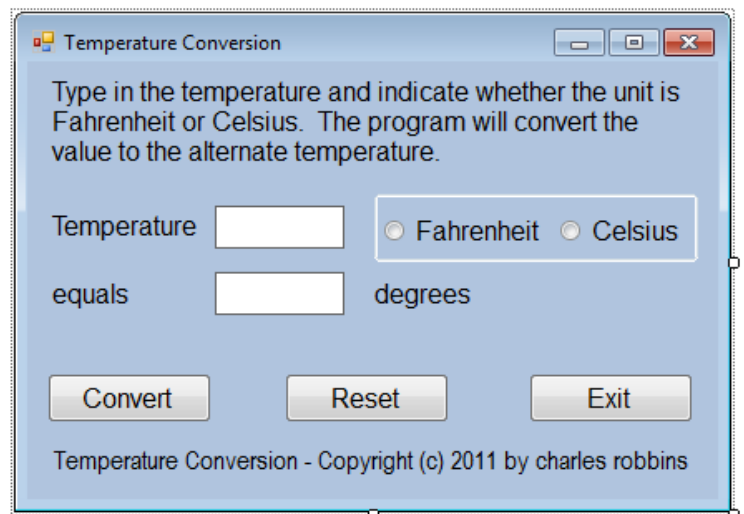


Figure 4B.20 – Adding a Copyright Statement

## Adding Comments in Visual C# to Communicate the Copyright

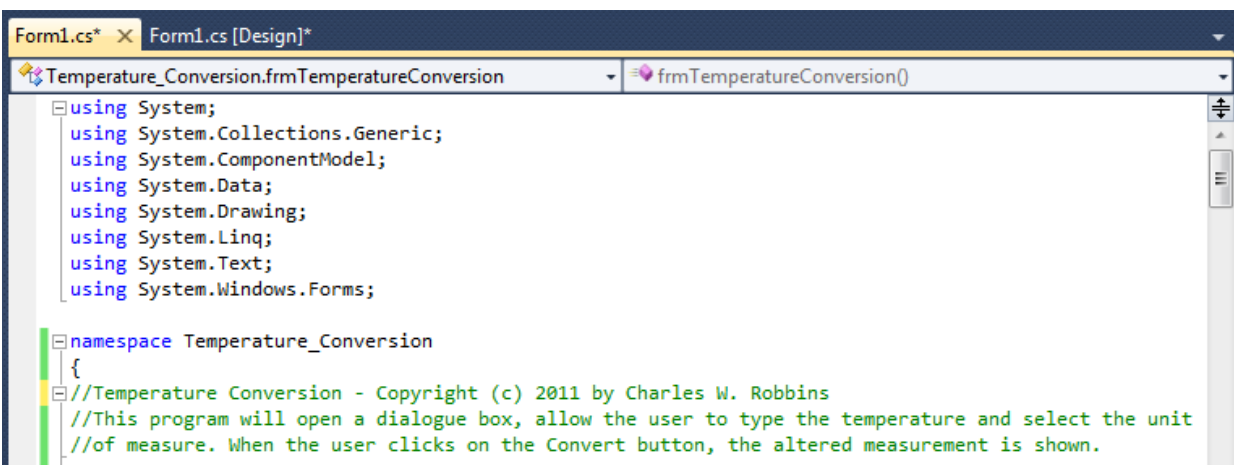
---

The comments we placed in the first three lines of the program will inform the individual opening and reading the code of the ownership. This is for those user that may run the application without checking the label on the bottom of the form with the copyright information. It is a great tool to alert the client to the rules of the program and tell them what the application will do.

To begin the actual coding of the program, double click on the form. At the top of the program and after the line of code with Namespace Temperature \_Conversion {, place the following comments with two slash (//) characters. Remember, the two slashes (//) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
//Temperature Conversion - copyright (c) 2011 by Charles W. Robbins  
//This program will open a dialogue box, allow the user to type the temperature and select the unit  
//of measure. When the user clicks on the Convert button, the altered measurement is shown.
```



```
Form1.cs* x Form1.cs [Design]*  
Temperature_Conversion.frmTemperatureConversion frmTemperatureConversion()  
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
  
namespace Temperature_Conversion  
{  
    //Temperature Conversion - Copyright (c) 2011 by Charles W. Robbins  
    //This program will open a dialogue box, allow the user to type the temperature and select the unit  
    //of measure. When the user clicks on the Convert button, the altered measurement is shown.
```

Figure 4B.21 – Adding a Copyright Statement

## Declaring Variables in a Program

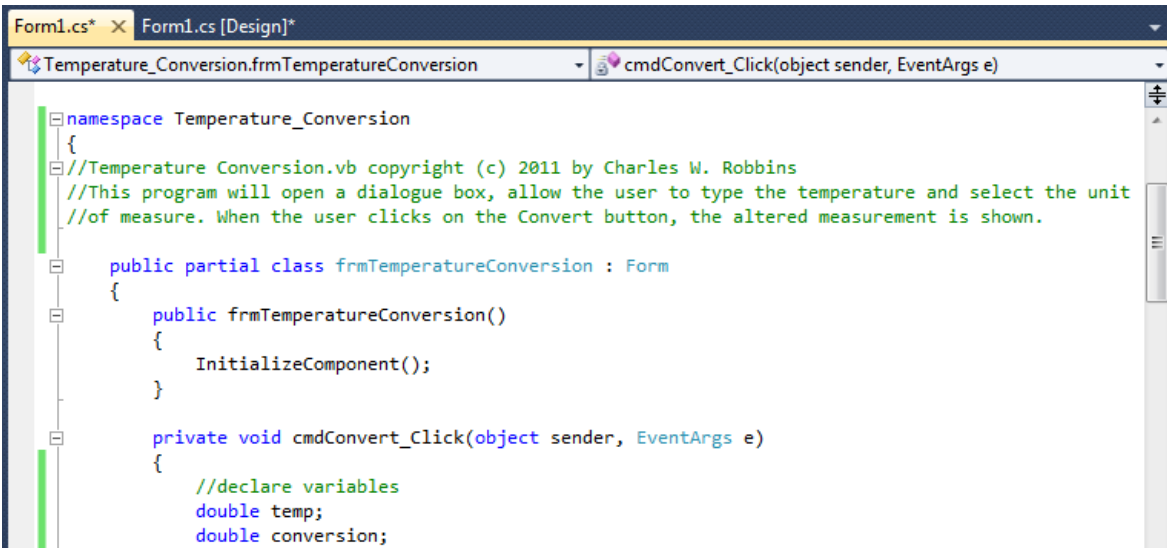
---

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In this Visual C# program, we will declare local variables in the cmdConvert subroutine.

In our program, we will retrieve the data from the textbox and also we will create data from mathematical computations. We will place the values in a variable called Temp. This variable will hold a number for calculations so we will declare it as a Double Integer.

Type the following code under the cmdConvert subroutine of the program.

```
//declare variables
Double temp;
Double conversion;
```



```
Form1.cs* x Form1.cs [Design]*
Temperature_Conversion.frmTemperatureConversion cmdConvert_Click(object sender, EventArgs e)
namespace Temperature_Conversion
{
    //Temperature Conversion.vb copyright (c) 2011 by Charles W. Robbins
    //This program will open a dialogue box, allow the user to type the temperature and select the unit
    //of measure. When the user clicks on the Convert button, the altered measurement is shown.

    public partial class frmTemperatureConversion : Form
    {
        public frmTemperatureConversion()
        {
            InitializeComponent();
        }

        private void cmdConvert_Click(object sender, EventArgs e)
        {
            //declare variables
            double temp;
            double conversion;
        }
    }
}
```

Figure 4B.22 – Declaring Variables with Dim Statements

Notice that the variable name should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the underscore character ( \_ ) to separate words in phrases. This is acceptable, but a double underscore ( \_\_ ) can cause errors if we do not detect the repeated character.

## Setting Variables in a Program

---

Next, we will set the variable using the equal function. We will set the number in the textbox to the variable **Temp**.

Type the following code under the "Set variable" section of the cmdConvert subroutine of the program.

```
//Set variable
temp = Convert.ToDouble(txtTemp.Text);
```

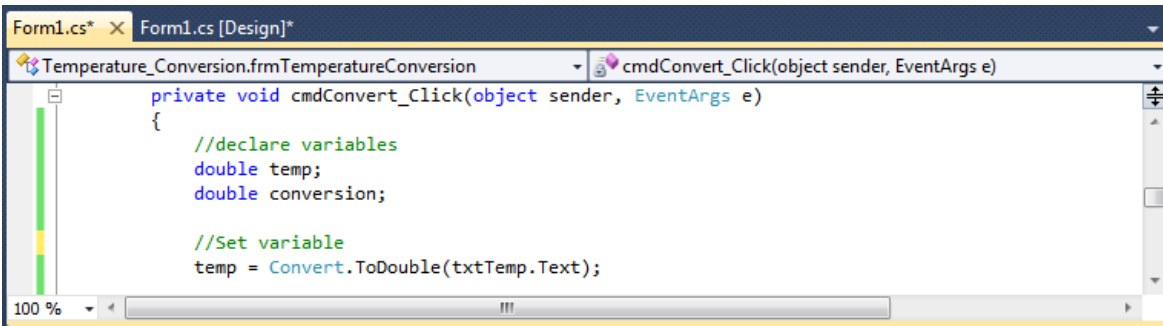


Figure 4B.23 – Setting the Variables

## Using a Condition Statement and Label to Communicate the Answer

---

Whenever we are confronted with making a choice between two or more options in a computer program, then the **if-then-else** function becomes a very popular solution to this challenge. The **if-then-else** function will execute the statements within the then section of the **if-then** expression when the logical test is true. The **if-then** function also will execute the **else** section of the **if-then** expression when the logical test is false.

The **if-then** function is arranged to work in a more complex fashion than other Visual C# tool. It is initially, and after that an expression containing the logical test is written right after the **if**. The logical expression tests for a true or false response. In this program, the test is whether the Fahrenheit radial button is true or checked. If the answer is true, then the label **lblAnswer** the calculated Celsius formula and the label **lblUnitAnswer** will equal **"degrees Celsius"**.

So type the following expression in the routine:

```
//Using a condition statement to get the answer
if (radFahrenheit.Checked == true)
{
    conversion = (temp - 32) * 5 / 9;
    lblAnswer.Text = conversion.ToString("F2");
    lblUnitAnswer.Text = "degrees Celsius";
}
```

In our **if-then-else** statement, we are going to use the **else** section of the function. If we needed to execute a statement for a false return, we would use the **else** section of the function. In this program, the test is whether the Fahrenheit radial button is true or checked. If the answer is false, then the label **lblAnswer** the calculated Fahrenheit formula and the label **lblUnitAnswer** will equal **"degrees Fahrenheit "**.

Go ahead and type the following the Else code below the condition statement.

```

//Using a condition statement to get the answer
if (radFahrenheit.Checked == true)
{
    conversion = (temp - 32) * 5 / 9;
    lblAnswer.Text = conversion.ToString("F2");
    lblUnitAnswer.Text = "degrees Celsius";
}
else
{
    conversion = 1.8 * temp + 32;
    lblAnswer.Text = conversion.ToString("F2");
    lblUnitAnswer.Text = "degrees Fahrenheit";
}

```

```

Form1.cs* × Form1.cs [Design]*
Temperature_Conversion.frmTemperatureConversion cmdConvert_Click(object sender, EventArgs e)
private void cmdConvert_Click(object sender, EventArgs e)
{
    //declare variables
    double temp;
    double conversion;

    //Set variable
    temp = Convert.ToDouble(txtTemp.Text);

    //Using a condition statement to get the answer
    if (radFahrenheit.Checked == true)
    {
        conversion = (temp - 32) * 5 / 9;
        lblAnswer.Text = conversion.ToString("F2");
        lblUnitAnswer.Text = "degrees Celsius";
    }
    else
    {
        conversion = 1.8 * temp + 32;
        lblAnswer.Text = conversion.ToString("F2");
        lblUnitAnswer.Text = "degrees Fahrenheit";
    }
}

```

Figure 4B.24 – Displaying the Answers

## Resetting the Data

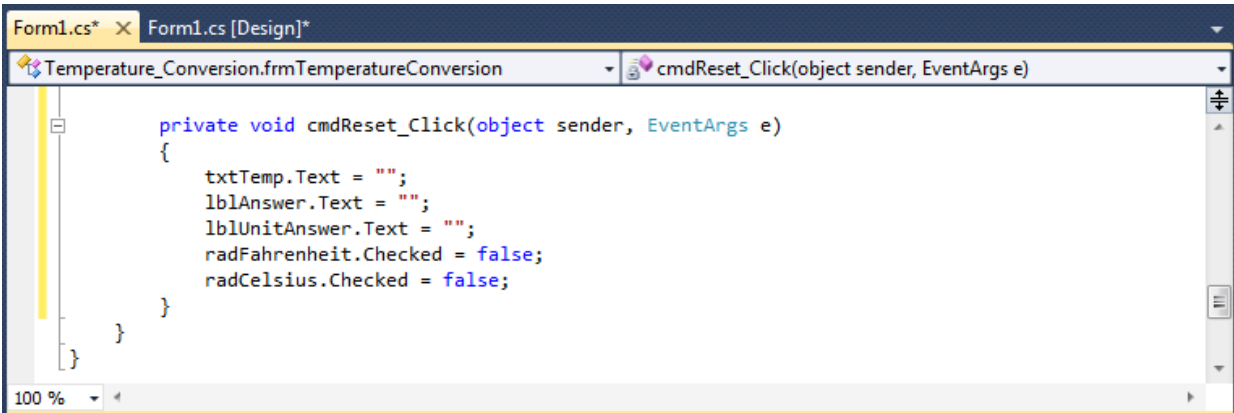
To clear the textboxes or labels containing the data, we will replace the date with blank strings and the date and time with the current day and time setting.

Type the following code under the cmdReset subroutine of the program



'Reset the textbox, the radial buttons and the labels

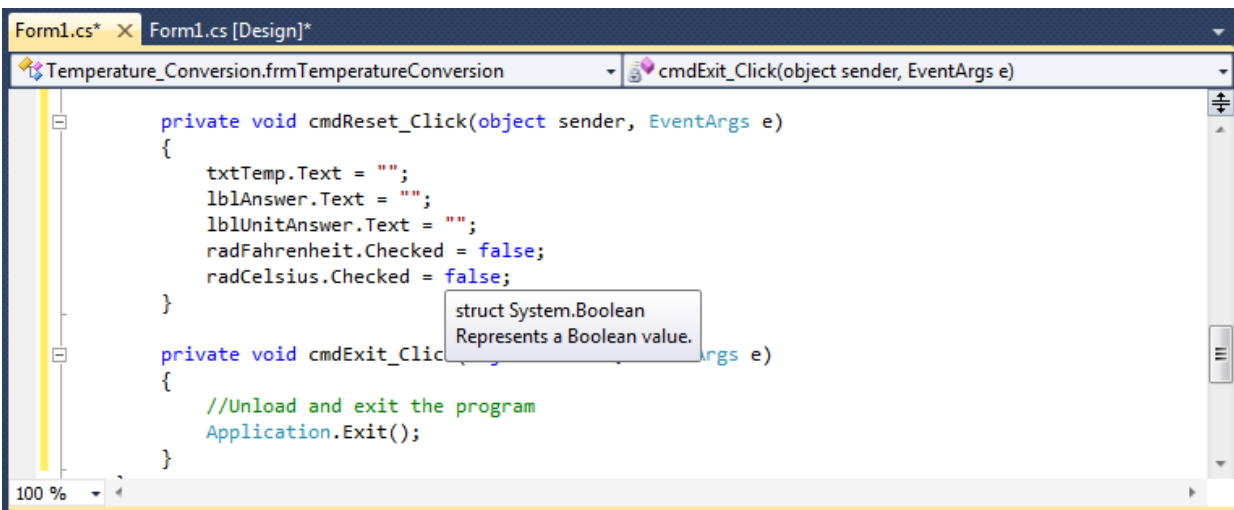
```
txtTemp.Text = ""  
lblAnswer.Text = ""  
lblUnitAnswer.Text = ""  
radFahrenheit.Checked = "false"  
radCelsius.Checked = "false"
```



```
Form1.cs* x Form1.cs [Design]*  
Temperature_Conversion.frmTemperatureConversion cmdReset_Click(object sender, EventArgs e)  
  
private void cmdReset_Click(object sender, EventArgs e)  
{  
    txtTemp.Text = "";  
    lblAnswer.Text = "";  
    lblUnitAnswer.Text = "";  
    radFahrenheit.Checked = false;  
    radCelsius.Checked = false;  
}  
}
```

Figure 4B.25 – Computing the Reset Button by Clearing a Textbox and Label Caption

## Exiting the Program



```
Form1.cs* x Form1.cs [Design]*  
Temperature_Conversion.frmTemperatureConversion cmdExit_Click(object sender, EventArgs e)  
  
private void cmdReset_Click(object sender, EventArgs e)  
{  
    txtTemp.Text = "";  
    lblAnswer.Text = "";  
    lblUnitAnswer.Text = "";  
    radFahrenheit.Checked = false;  
    radCelsius.Checked = false;  
}  
  
private void cmdExit_Click(object sender, EventArgs e)  
{  
    //Unload and exit the program  
    Application.Exit();  
}  
}
```

Figure 4B.26 – Exiting the Program

To exit this program, we will unload the application and end the program.

Type the following code:

```
//Unload and exit the program  
Application.Exit();
```

## Running the Program

After noting that the program is saved, press the F5 to run the Temperature Conversion program. The Temperature Conversion window will appear on the graphical display as shown in Figure 4B.27. Notice the professional appearance and presentation of information in a clean dialogue box.

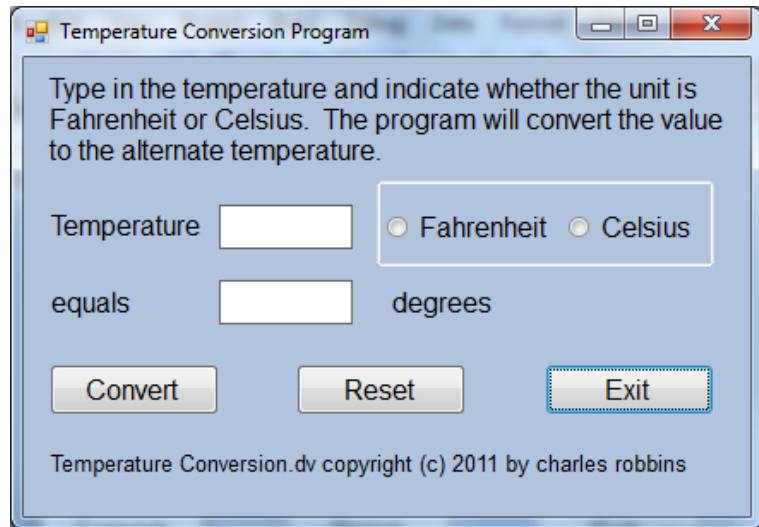


Figure 4B.27 – Launching the Program

Type the Fahrenheit temperature in the textbox just as we typed as shown in Figure 4B.28. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Convert command button and the two answer labels will have the numeric quantity and the unit of measure.

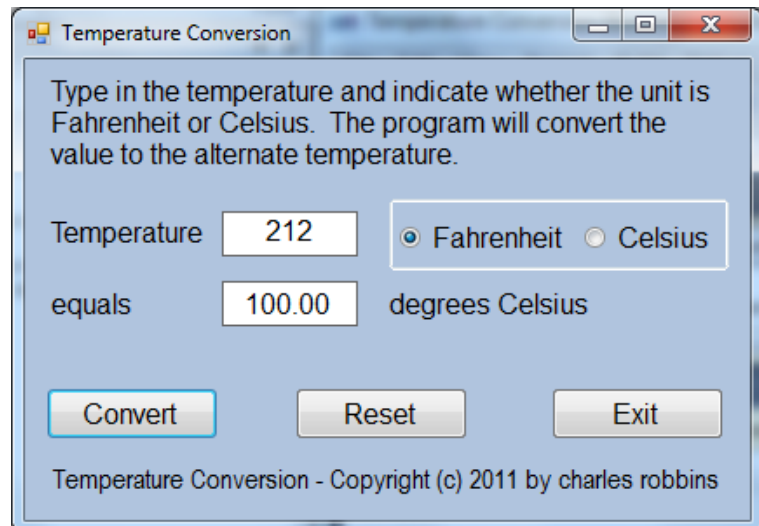
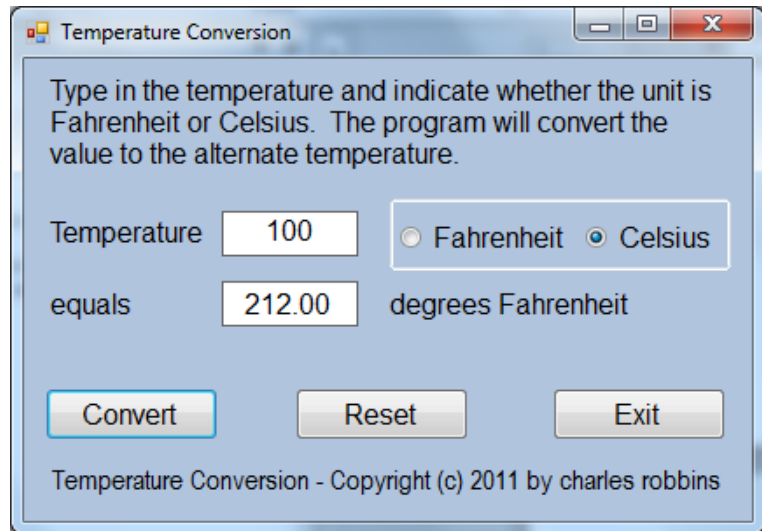


Figure 4B.28 – Running the Program

Type the Celsius temperature in the textbox just as we typed as shown in Figure 4B.29. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Convert command button and the two answer labels will have the numeric quantity and the unit of measure. After experimenting with our program, press the Exit command button to exit the application.



**Figure 4B.29 – Running the Program a Second Time**

If our program does not function correctly, go back to the code and check the syntax against the program shown in previous sections. Repeat any processes to check or Beta test the program. When the program is working perfectly, save and close the project.

There are many variations of this Visual C# Application we can practice and obtain information from a personal computer. While we are practicing with forms, we can learn how to use variables, strings and comments. These are skills that we want to commit to memory.

**\* World Class CAD Challenge 90-3B \* - Write a Visual C# Application that displays a single input form, allows the user to type in their data, and when executed, the program will give the user information obtained from the computer and from mathematical computations.**

**Continue this drill four times using some other form designs, each time completing the Visual C# Project in less than 1 hour to maintain your World Class ranking.**