Chapter 5

Visual C#: Condition Statement

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

- **Writing Condition Statements in Visual C#**
- **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 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 Label to Communicate with Variables**
- **Determining a String Length with the Function**
- **Using a Loop with the While Function**
- **Remove a Character from a String**
- **Changing a Character to ASCII**
- **Testing a Case with the If -Then Function**
- **Adding One to the Counter in the Loop**
- **Using More If-Then Functions**
- **Resetting the Data**
- **Exiting the Program**
- **Running the Program**

Writing Condition Statements in Visual C#

When a person comes to an intersection in a road, they have a decision to make whether to turn right or to turn left or maybe just go straight ahead. In programming, we need to make the same type of decision based upon a test. Back at the road, if we wanted to go to the store, possibly we would turn to the right. If we wanted to visit a friend, then maybe we will choose the left turn or finally we want to head on home and we go straight. Back to the program, in this chapter, we will learn how to write if-then statements to test whether a password meets the complexity level and the string length to meet the strong password criteria.

The definition of a strong password is that the string of characters needs to meet three of the following conditions. At least one character needs to be uppercase. At least one character needs to be lower case. At least one character needs to be a number. At least one character should be a special character such as question mark, exclamation mark, dash or And sign (?, !, -, &). The password should be at least seven characters in length. Meeting the length requirement and three out of the four types of characters conditions make the string of text a strong password.

A method that we can use in doing the test is to remove a single character at a time from the password string and make four tests, one for each condition. By passing a single test, we will change a variable named for that particular test to a "1" or On state. For example, if our password is "WorldClassCAD" and we remove the capital **W**, in one test we would say if the **W** falls between the capital A and capital Z, then we will change the variable named Uppercase from zero to one. Below, we show an example of this code.

```
'Test for uppercase
If holder>=65 and holder<= 90 then
        uppercase = 1
End If
```

We will test all four conditions on each letter, knowing that only one condition will change. As we continue with our testing in another example, we will test the next letter, the **o** from the password "WorldClassCAD". In one test for the letter **o**, we would say if the **o** falls between the lowercase a and lowercase z, then we will change the variable named Lowercase from zero to one. As we continue testing through the password, we can see only the Uppercase and Lowercase variables will change to the On state and the other two variables, Special and Number will remain zero or Off. Even though the password "WorldClassCAD" has thirteen characters and is strong in length, however this password is not strong because it has only met two out of four character conditions.

We will use **if-then** statements to construct the text to announce whether the string of characters does or does not meet the strong password criteria. If we find a password to be lacking in one or more conditions, we will use the same type of decision statement to construct a sentence like **"Make at least one character uppercase".** We will concatenate sentences together then broadcast them using a message box. This program is practical and helps us to develop specialized skills and since there are multiple lines of code in this project that gives us a better chance to remember this useful skill. The **if-then** statement will become a cornerstone function and every one of our programs throughout a career will likely use this function.

We still will use message boxes, declare variables, and assign values to variables just as we did in the previous chapters. We want to start the learning process using the skills we learned in chapter 2 and 3, so in this chapter, we will start first by using those message and input boxes. So let's get started.

Open the Visual C# Editor

In this lesson, we will step through each procedure in adding labels, textboxes and command buttons and we will integrate them into the tutorial along with condition statements, a while loop and message boxes. As in every project, we will create variables, set their values, use functions to manipulate the data and output data.

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

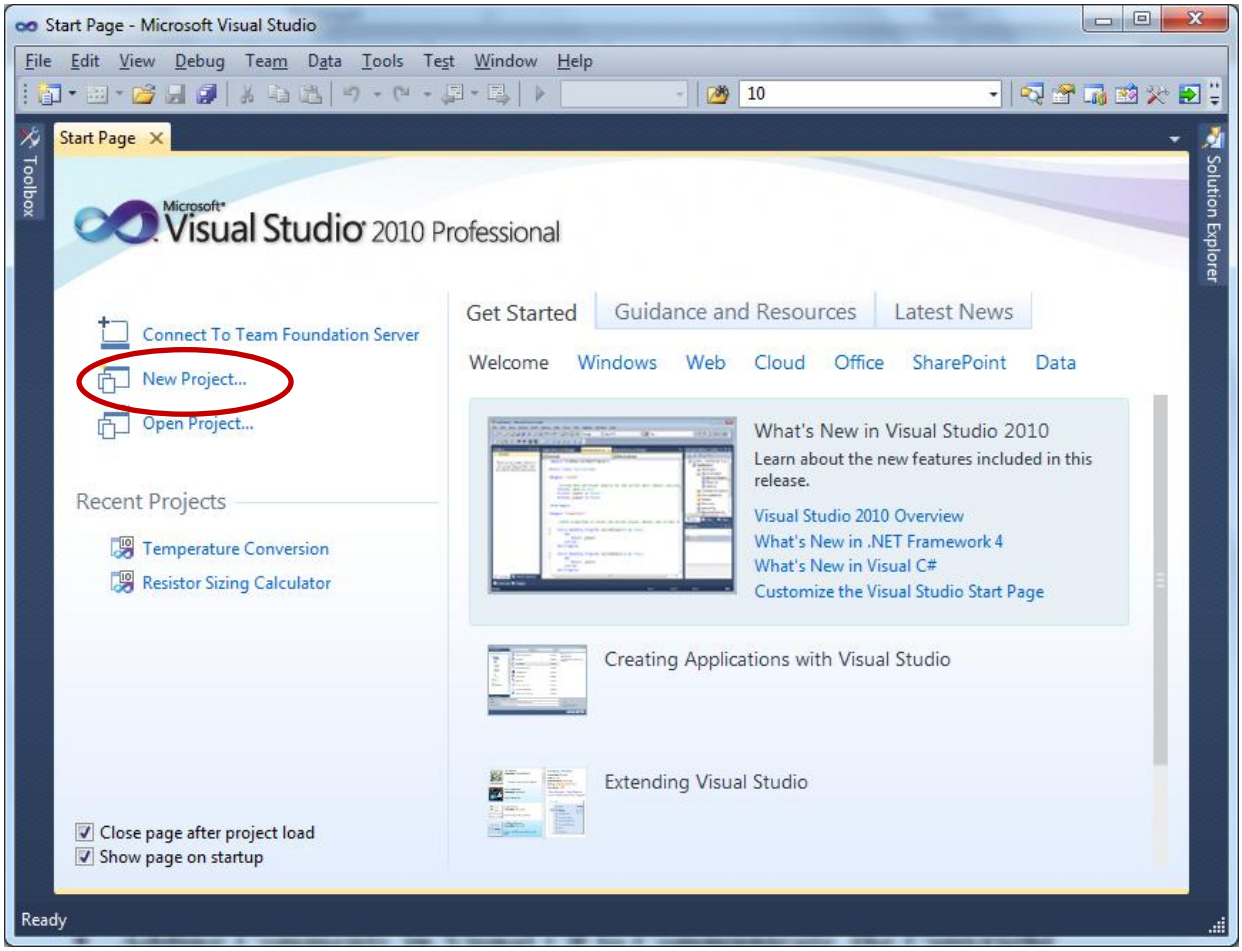

Figure 5.1 – The Start Page

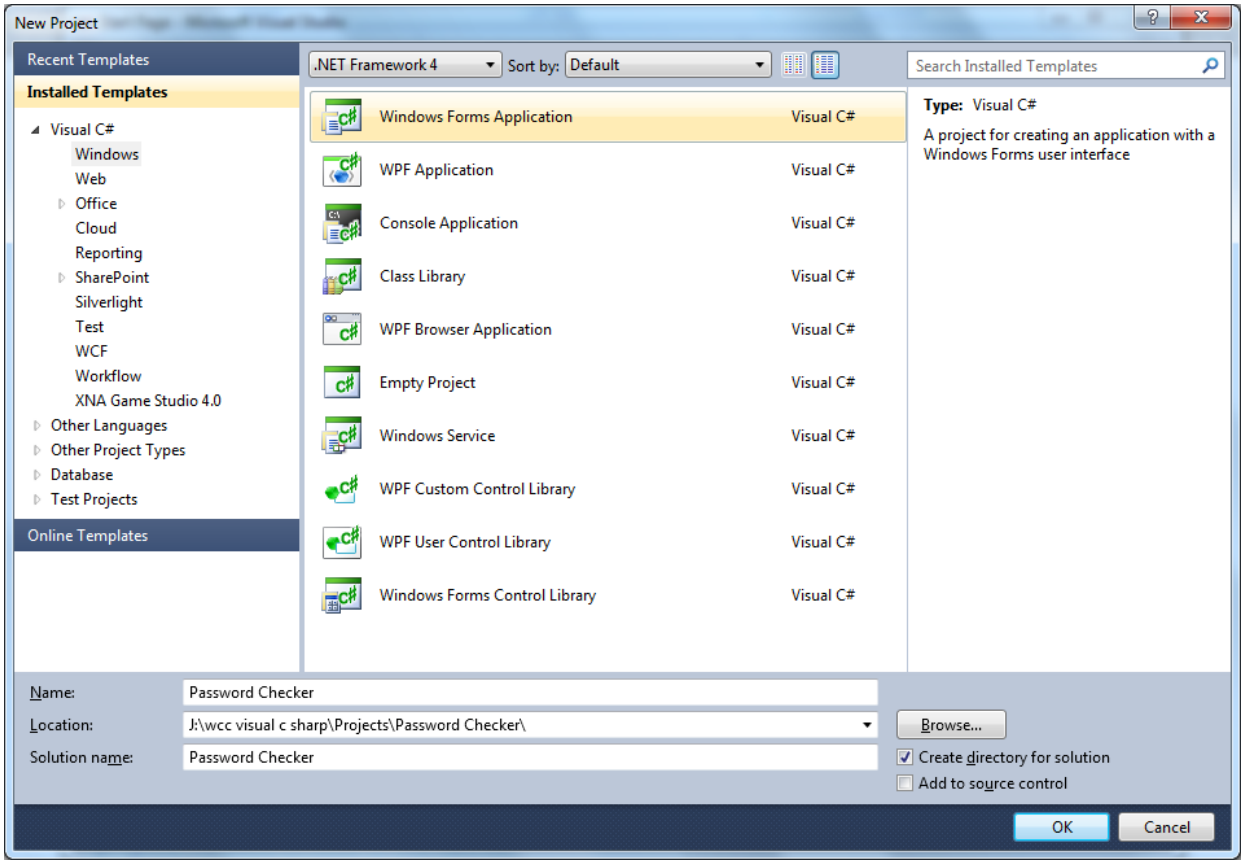

Figure 5.2 – New Project

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, Password Checker. 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 Password Checker. On the New Project window, we browse to the Password Checker 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, and command buttons. In this project, we will name the input form, Password Checker. We will have a label that explains the strong password rules. We will have a textbox to key in the password. We will have three command buttons, Check, 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.

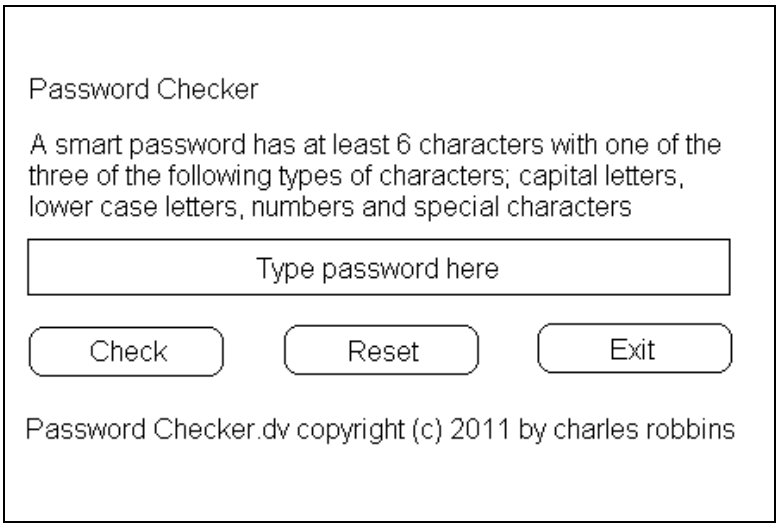

Figure 5.3 – Sketch of the Password Checker Form

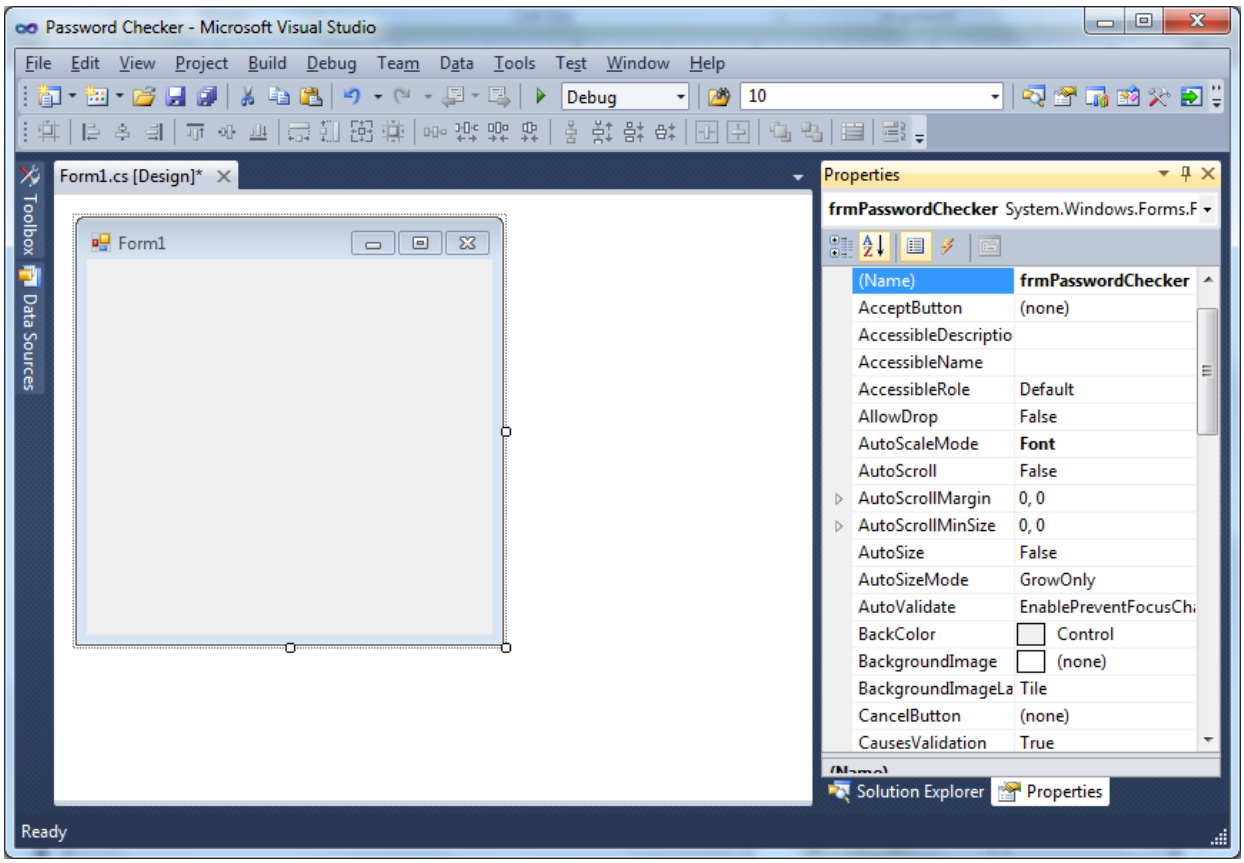

Figure 5.4 – Designing the Password Checker Form in Visual C#

We will change the **Text** in the Properties pane to Password Checker to agree with the sketch in Figure 5.3. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
Font	Arial, 16 pt
Size	600, 300

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

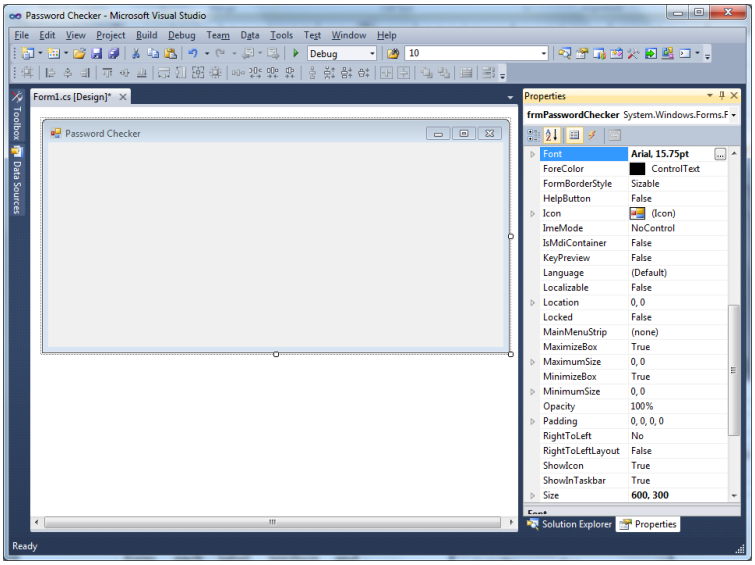

Figure 5.5 – Setting the Form Properties

The background color will change to a white. 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.

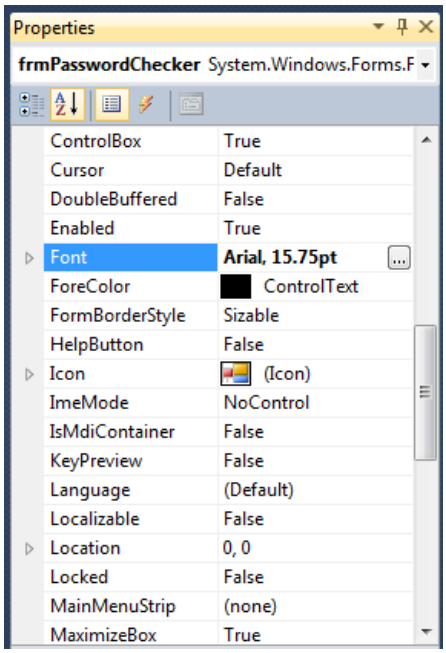

Figure 5.6 – The Font Window in Visual C#

We will select the Arial font, Regular font style and 16 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.

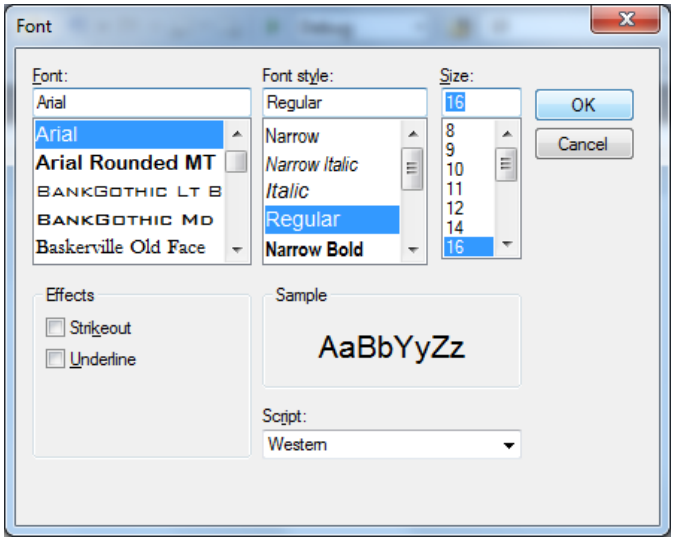

Figure 5.7 – Changing the Font to Arial

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 **lblRule.**

Alphabetic	
(Name)	lblRule
AutoSize	False
Text	A smart password has at least 6 characters with one of the three of the following types of characters; capital letters, lower case letters, numbers and special characters

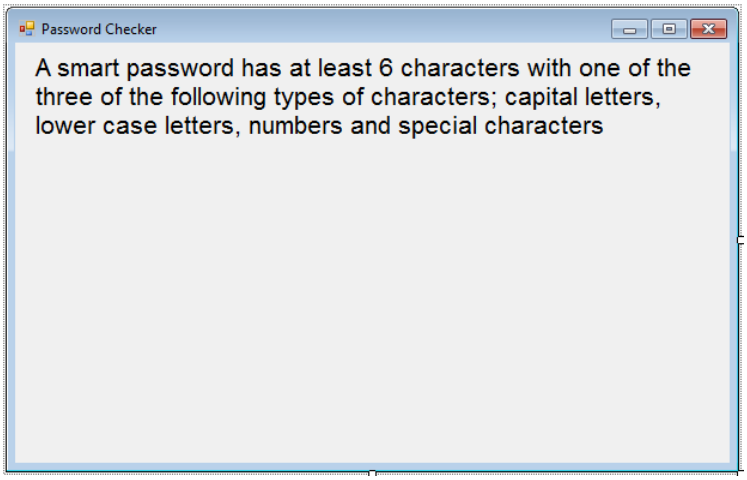

Figure 5.8 – The Finished Label on the Form

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.

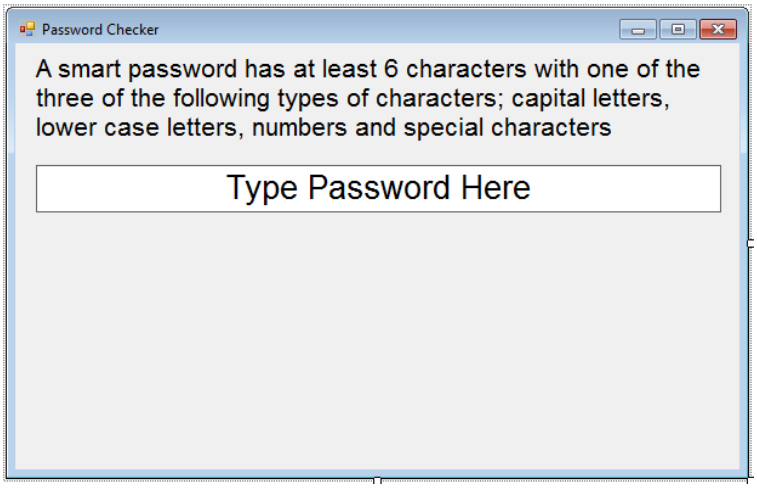

Figure 5.9 – Placing a TextBox on the Form

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 **txtPassword.**

Alphabetic	
(Name)	txtPassword
Size	556,39
Text	Type Password Here
TextAlign	Center

The size of the textbox will be 560 wide and 39 tall and the characters inside the textbox will be aligned in the center.

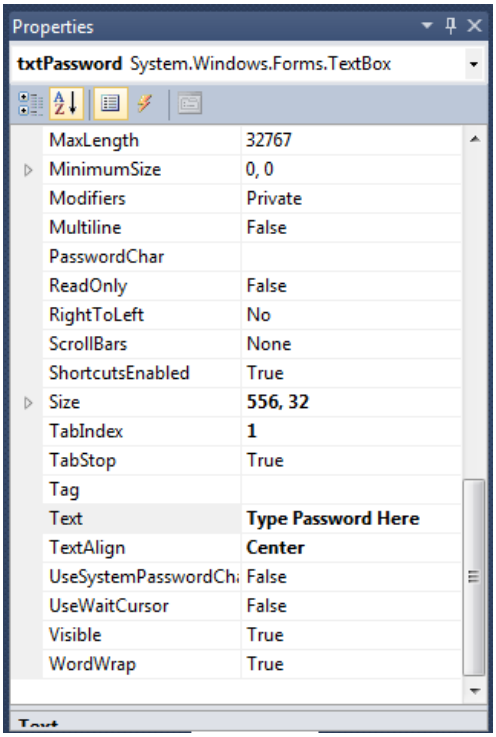

Figure 5.10 – Setting the Size of the Textbox

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.

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

Alphabetic	
(Name)	cmdCheck
Caption	Check
Font	Arial, 15.75 pt
Size	127,45

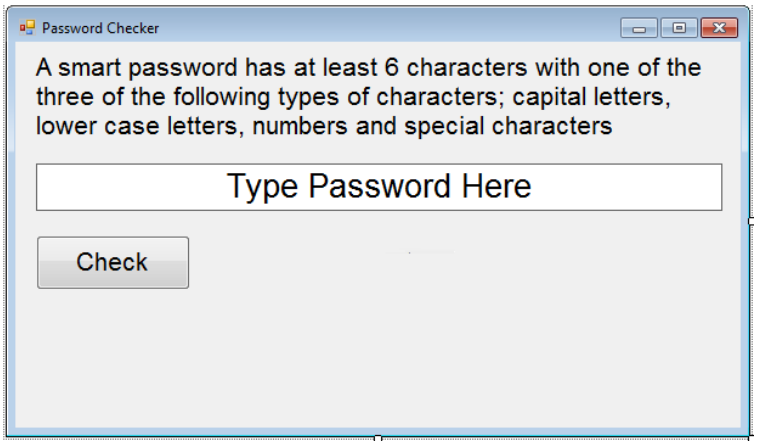

Figure 5.11 – The Command cmdCalculate Button

Add a second Command button; named cmdReset is for clearing the txtPassword object. 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.

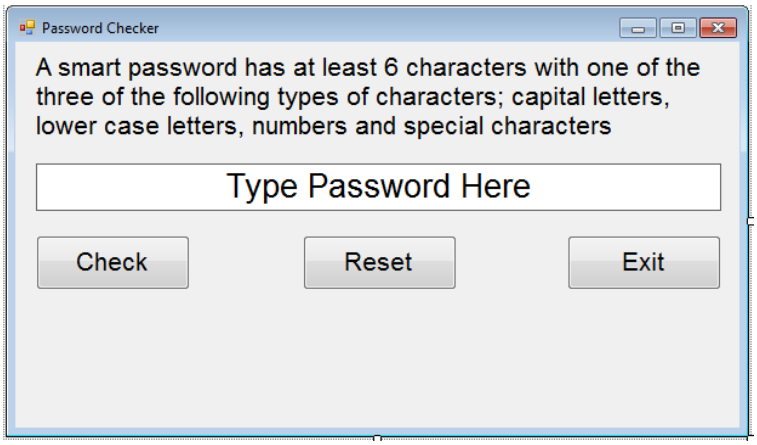

Figure 5.12 – Insert Two More Command Buttons

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.

Password Checker - 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.

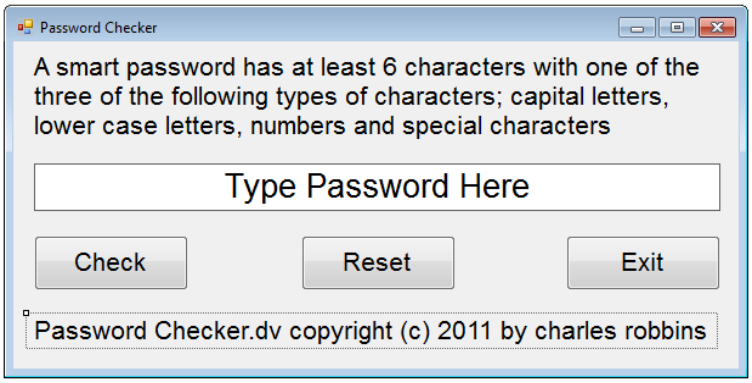

Figure 5.13 – Adding a Copyright Statement

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, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of the program and what will the application do.

To begin the actual coding of the program, double click on the Hello command button. At the top of the program and before the line of code with Private Sub cmdCheck_Click (), place the following comments with the single quote (‘) character. Remember, the single quote character (‘) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
//Password_checker - Copyright (c) 2007 by Charles W. Robbins
//this program will check the password against complexity requirements and length
```

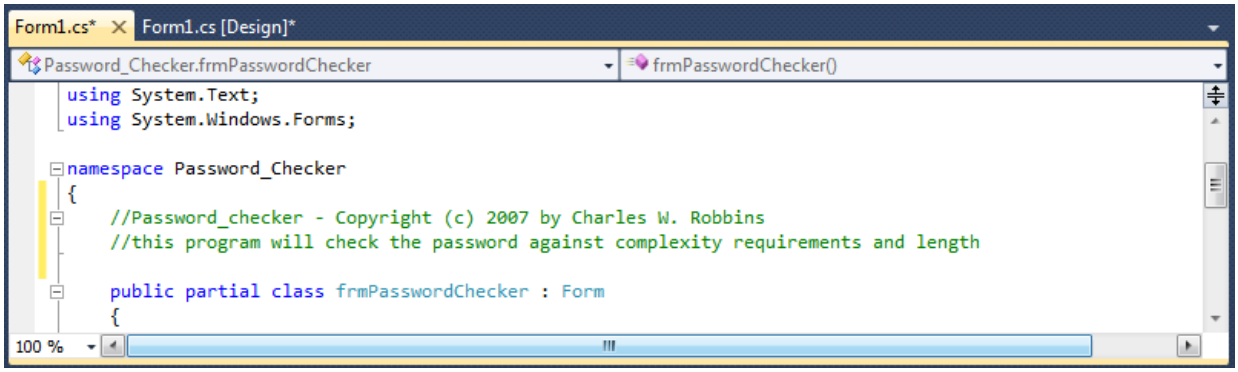

Figure 5.14 – Adding a Copyright Statement

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 a local variable in the cmdCheck subroutine.

In our program, we will retrieve the data from the textboxes and also we will create data from mathematical computations. These variables will hold numbers for calculations so we will declare them as Integers.

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

```
//Declare variable
  int counter;
  int holder1;
  int uppercase;
  int lowercase;
  int number;
  int special;
  int pwdlength;
  int check;
  string password;
  string character;
  string msg1;
  string msg2;
  string msg3;
  string msg4;
  string msg5;
```

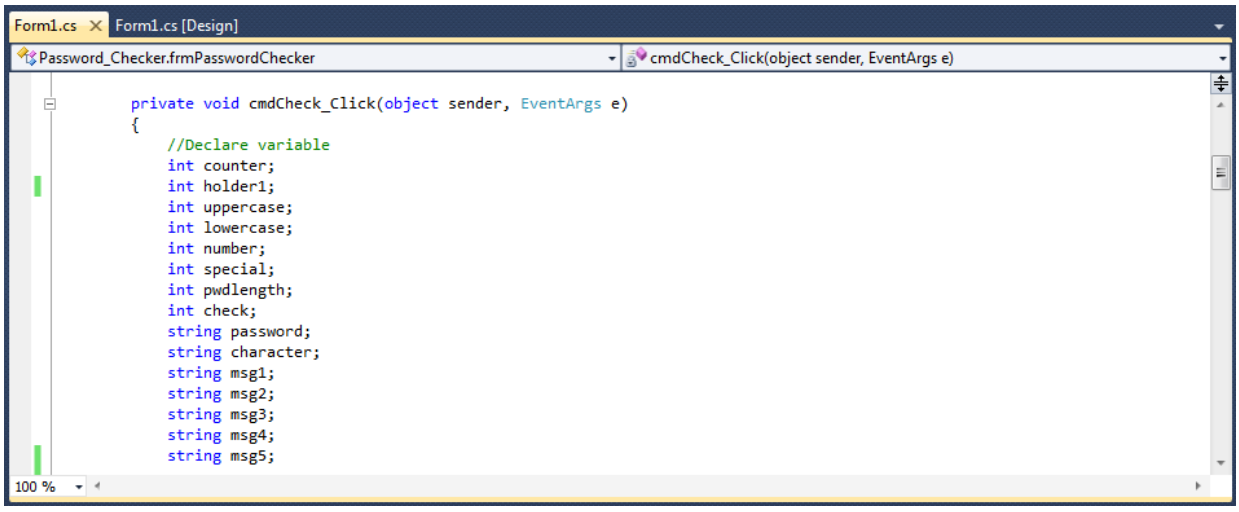

Figure 5.15 – Declaring Variables with Dim Statements

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 variables using the equal function. We will set the characters in the textbox to the password variable and we will set the number, uppercase, lowercase, special and counter to zero. We also assign null values to the strings msg1 through msg5.

Type the following code under the "set variable" section of the cmdCheck subroutine of the program.

```
//Set variables
    password = txtPassword.Text;
    number = 0;
    uppercase = 0;
    lowercase = 0;
    special = 0;
    counter = 0;
    msg1 = "";
    msg2 = "";
    msg3 = "";
    msg4 = "";
    msg5 = "";
```

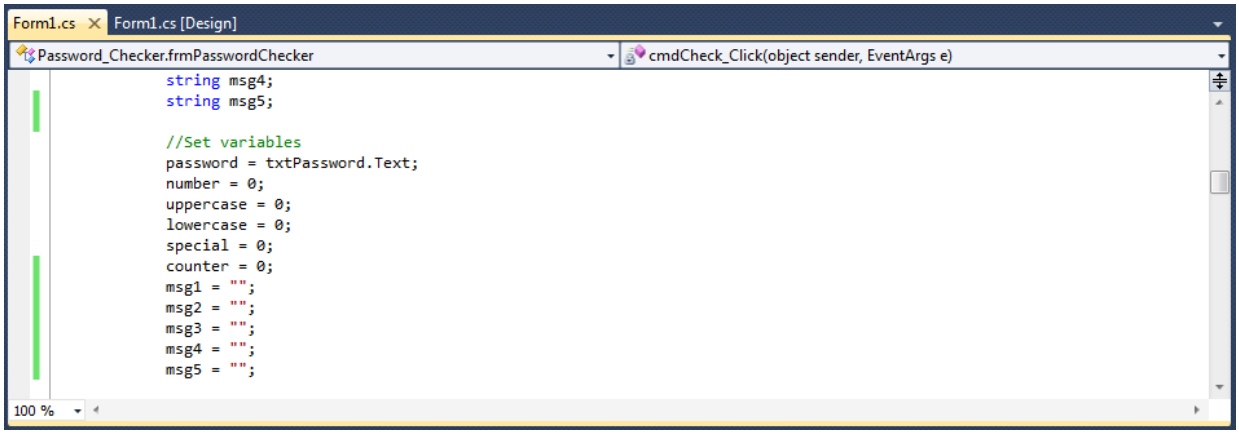

Figure 5.16 – Setting the Variables in the VBA Code

Determining a String Length

When we want to find how many characters are in a text string then we will use the length function. In a simple case, like with the word "test", the answer the "test".length is 4. When there is a space in the text string, like "easy coding" then "easy coding".length is 11. The space counts in the computation of the text string length. Type the following code in the program.

```
//Determine password length
pwdlength = password.Length;
```

Using a Loop with the While Function

Many years ago I brought a class of students through the steps of creating while loops in their computer programs. In that exercise, I had the students create a basement stairs completely from scratch using a visual program. The problem involved some mathematics, the knowledge of selections sets, and of course the while loops. I would have to say that most of the students really struggled through the exercise with me. My approach was too difficult. My challenge was to find a technique to train powerful programming functions and simultaneously allowing the programming student to concentrate on coding.

The next group going through the lesson plan for while loops at the college, I still used a step with a run of 10 inches in a rise of the 8 inches. We repeated the single step ten times to construct a simple looking stairs. We drew a ball and bounced it down the stairs using the while loop. They enjoyed the simplicity of the assignment and went on to make very nice looking animations. In our first while loop in Visual C#, we are going to remove a single letter and test the character four times. Sounds pretty effortless and through simplicity we learn how to use another useful tool.

When we are using a Do While Loop function in a Visual C#, right after the words Do While we will place a test statement that will be used by the Do While function each time to determine whether to enter or exit the loop. In our example and in most of our programs, we will use the counter. We set the variable counter previously to zero. We know the number of characters in the password which is held in the variable "length". The test is simple. We will stay in the loop as long as the variable counter is less than length.

If the password is 13 characters long, the first time into the loop the condition is (< 0 13) which is true so all of the expression inside the while loop will be read in the program. The second time into the loop the condition is (< 1 13) which is also true so all the loop continues to run. The third time into the loop the condition is (< 2 13) which is also true and the loop continues. The fourth time into the loop the condition is (< 3 13) which is also true and this seems to be going on and on.

In a classroom we go through every step on our first while loop. By this time many students do not think this will ever end. The thirteenth time into the loop the condition is (< 12 13) which is

also true, because 12 is less than 13. Now on fourteenth time into the loop the condition is (< 13 13) which is false and so the while loop will not execute and the next expression in the code will be read.

Start the while loop by typing the following.

```
while (counter < pwdlength)
{
```

Remove a Character from a String

Once inside the loop, we need to extract a single letter at a time from the password in order to test the letter against the strong password criteria. We will do this operation by using the **left** function. The left function is set up as shown below.

```
//Check password
character = password.Substring(0, 1);
```

We type the name **character**, which is a variable name that will hold a single character and then an equal sign. After entering the variable name password which hopefully contains seven or more characters, we type a dot and the substring function. Then comes an open parentheses. Now to retrieve the first character, we will type zero in the first character will be taken from the password. The second number (1) is the string length. The line of code ends with a closed parentheses.

Someday, we may want to use this function for other purposes so we should know that we can modify this same line of code to

```
character = password.substring(0,3)
```

And then the variable **character** will contain the first three letters of the password text string.

Changing a Character to ASCII

Another easy function to learn to use is the ASCII tool. We named the variable holder1 because we are just translating the keyboard character extracted from the password to a numeric code. Americans Standard Code for Information Interchange or ASCII format has the number that designation for every keyboard character. Type the following code.

```
char c = character[0];
holder1 = System.Convert.ToInt32(c);
```

The reason we are changing the extracted character from the password to a numeric value is so that we can be easily compare this translated value to a range of numeric values that represent uppercase letters, lowercase letters, numbers and special characters. In the ASCII appendix of

this text, we can see that the numeric range of uppercase letters is from 65 to 90. In the same table, we see that numeric arrange for lowercase letters to be from 97 to 122. Again, the table shows the range zero to nine on the keyboard to be from 48 to 57. For special characters, we will use an exact numbers to check the single extracted character. This is a very easy technique to use in programming.

Testing a Case with the If -Then Function

Whenever we are confronted with making a choice between two or more options in a computer program, then the **if-then** function becomes a very popular solution to this challenge. The **if-then** 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# tools. If 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 ASCII number in the variables **holder1** is greater or equal to 65 and less or equal to 90. If the answer is true, then the variable **uppercase** is assigned the value of one. If the answer is false, then the variable **uppercase** remains a zero as assigned earlier in the program.

So type the following expression in the routine:

```
 //Test for uppercase
if (holder1 >= 65)
{
   if (holder1 <= 90)
   {
      uppercase = 1;
   }
}
```

In our first **if-then** statement, we are not 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, we get to practice our first **if-then** statements another three times. Type the following lines of code to test the single extracted character for lowercase, number and special character conditions.

The test case in the **if-then** statement has an **And** function. The test case is

holder>=65 and holder<= 90

where the ASCII number holding variable has to be greater than or equal to 65 and less than or equal to 90. When we need an And function, we write two if then statements in succession.

```
//Test for lowercase
if (holder1 >= 97)
{
   if (holder1 <= 127)
   {
      lowercase = 1;
   }
}

//Test for number
if (holder1 >= 48)
{
   if (holder1 <= 57)
   {
      number = 1;
   }
}

//Test for special character
if (holder1 == 21 || holder1 == 36 || holder1 == 45 || holder1 == 63)
{
   special = 1;
}
```

In the test case in the **if-then** statement has an **Or** function. The test case is

```
if (holder1 == 21 || holder1 == 36 || holder1 == 45 || holder1 == 63)
```

where the ASCII number holding variable has to be equal to 21 or 36 or 45 or 63. When we need an Or function, we write two || vertical bars in the if then statements as shown.

Removing Single Character at a Time

The **mid** function will work wonders for us if we know how to use this text handling function.

By typing a comma and a number after the variable **password,** we can extract just about any part of a text string that we wish. In this case we will extract all the characters after the first character. When we leave the next integer or whole number off after the 2 in the line of code:

```
password = password.Substring(1);
```

then all remaining characters are extracted

Adding One to the Counter in the Loop

Now, we need to discuss the properties of a programming loop. The easiest technique we have demonstrate this concept is to take a small group of students and form a circle. We identify the beginning of the circle to be one person and we give them the value of 0 and they say out load "zero". Each person passes the marker around clockwise until a student hands the first person the same marker back. Now in order to make a basic programming loop function properly, we will add one to the beginning value, so the first person announces "one" and continues to pass the marker around the group again. The next time the first person receives the marker they have figured out the game by now, and states "two". To allow this to become common place in their programming skill set, typically while we are discussing the writing of the **Adding** expression, we allow them to continue until the entire group is tired of passing the marker in a circle and hearing the count by ones, but finally someone will ask how to stop the loop. That will be another discussion with another function, but remember where you heard about loops first, using the **adding** function. Okay, we can stop counting, now.

The next expression we will place in the code will add one to the variable **counter**. We will use the adding function to add 1 to the counter, so the variable **counter** will now be 1.

```
counter = counter + 1;
```

At the end of the Do While loop type Loop. At this point of the program the computer will return to the initial starting point and the test in the Do While case will be run again. The loop will continue until the case is false.

Using More If-Then Functions

After exiting the Do While loop, we will check the password to see if the text string meets the four criteria. Type the following code as shown.

```
//Check the password
check = uppercase + lowercase + number + special;
```

Our strategy is to have the variable check equal to 3 or more, which means that the password has met the 3 out of 4 types of characters.

Next, we will test for the password length. We captured his piece of information earlier before entering the loop. If the text string is less than seven characters long, then we will construct a message recommending at least a seven character text string.

```
//Check the password for length
check = uppercase + lowercase + number + special;
```

```
if (pwdlength < 7)
{
   msg1 = "Make the password at least 7 characters long. ";
}
```

We will now check each value of the test variables to see if they are equal to zero and if they are, we will make a recommendation to change the password for this consideration.

Type the following code where we will assign a sentence to a message named msg2, msg3, msg4 and msg5. We will use the messages in the last message box informing the user of the results of the program.

```
//Check the password for uppercase letters
if (uppercase == 0)
{
   msg2 = "Make at least one character uppercase. ";
}
//Check the password for lowercase letters
if (lowercase == 0)
{
   msg3 = "Make at least one character lowercase. ";
}
//Check the password for numbers
if (number == 0)
{
   msg4 = "Make at least one character a number. ";
}
//Check the password for special characters
if (special == 0)
{
   msg5 = "Make at least one character a special character. ";
}
```

In the last if-then statement, we will use and else section to announce that the password is not strong and to construct the messages from the previous section to what changes should be made. In this statement the case is

check>=3 and length>=7

and if the answer is True then the

"Password is strong"

Type the following code as shown.

```
if (check >= 3)
{
   if (pwdlength >= 7)
   {
      MessageBox.Show("Password is strong");
   }
}
```

If the answer is not true then we use the Else function to develop our message.

```
else
   {
      MessageBox.Show("Password is not strong. " + msg1 + msg2 + msg3 + msg4 + msg5);
   }
}
```

We will use + to concatenate the four test messages to the first sentence "Password is not strong."

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 password textbox
txtPassword.Text = "Type Password Here"
```

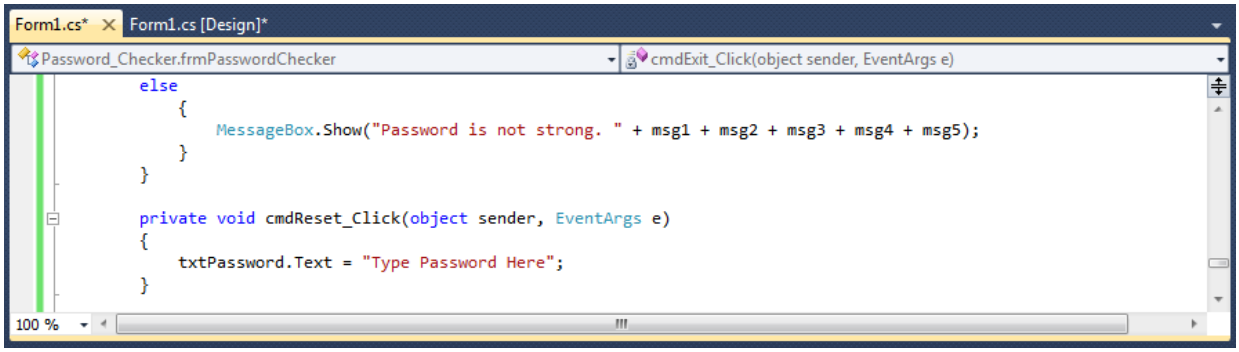

Figure 5.17 – Computing the Reset Button by Clearing a Textbox

Exiting the Program

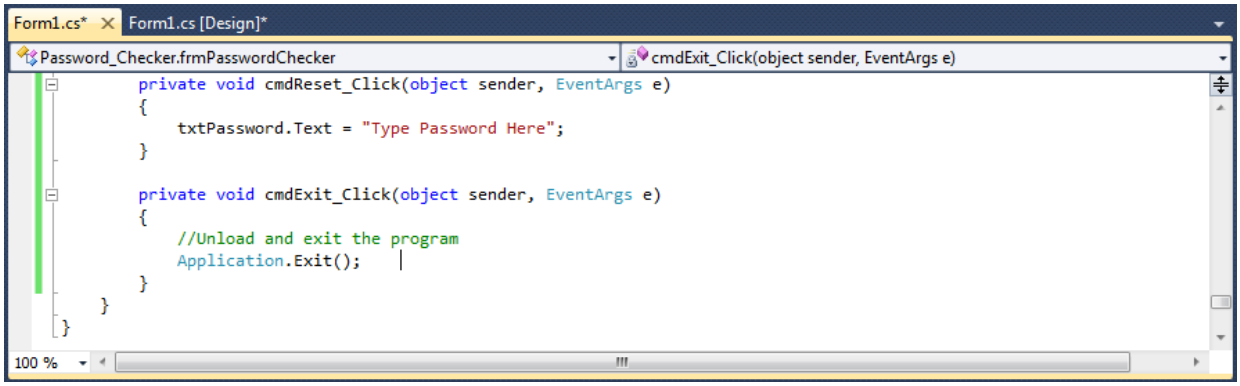

Figure 5.18 – Exiting the Program

To exit this program, we will unload the application and end the program.
Type the following code:

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

Written below is the entire Password_checker.vbs code for your benefit.

```
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 Password_Checker
{
   //Password_checker - Copyright (c) 2007 by Charles W. Robbins
   //this program will check the password against complexity requirements and length

   public partial class frmPasswordChecker : Form
   {
      public frmPasswordChecker()
      {
         InitializeComponent();
      }
```

```
private void cmdCheck_Click(object sender, EventArgs e)
{
   //Declare variable
   int counter;
   int holder1;
   int uppercase;
   int lowercase;
   int number;
   int special;
   int pwdlength;
   int check;
   string password;
   string character;
   string msg1;
   string msg2;
   string msg3;
   string msg4;
   string msg5;

   //Set variables
   password = txtPassword.Text;
   number = 0;
   uppercase = 0;
   lowercase = 0;
   special = 0;
   counter = 0;
   msg1 = "";
   msg2 = "";
   msg3 = "";
   msg4 = "";
   msg5 = "";

   //Determine password length
   pwdlength = password.Length;

   while (counter < pwdlength)
   {

   //Check password
   character = password.Substring(0, 1); ;

   char c = character[0];
   holder1 = System.Convert.ToInt32(c);
```

```
    //Test for uppercase
    if (holder1 >= 65)
    {
        if (holder1 <= 90)
        {
            uppercase = 1;
        }
    }

    //Test for lowercase
    if (holder1 >= 97)
    {
        if (holder1 <= 127)
        {
            lowercase = 1;
        }
    }

    //Test for number
    if (holder1 >= 48)
    {
        if (holder1 <= 57)
        {
            number = 1;
        }
    }

    //Test for special character
    if (holder1 == 21 || holder1 == 36 || holder1 == 45 || holder1 == 63)
    {
        special = 1;
    }

    password = password.Substring(1);
    counter = counter + 1;
    }

//Check the password for length
check = uppercase + lowercase + number + special;

if (pwdlength < 7)
{
    msg1 = "Make the password at least 7 characters long. ";
}
```

```
        //Check the password for uppercase letters
        if (uppercase == 0)
        {
            msg2 = "Make at least one character uppercase. ";
        }
        //Check the password for lowercase letters
        if (lowercase == 0)
        {
            msg3 = "Make at least one character lowercase. ";
        }
        //Check the password for numbers
        if (number == 0)
        {
            msg4 = "Make at least one character a number. ";
        }
        //Check the password for special characters
        if (special == 0)
        {
            msg5 = "Make at least one character a special character. ";
        }

        if (check >= 3)
        {
            if (pwdlength >= 7)
            {
                MessageBox.Show("Password is strong");
            }
        }
        else
            {
                MessageBox.Show("Password is not strong. " + msg1 + msg2 + msg3 + msg4 + msg5);
            }
        }

        private void cmdReset_Click(object sender, EventArgs e)
        {
            txtPassword.Text = "Type Password Here";
        }

        private void cmdExit_Click(object sender, EventArgs e)
        {
            //Unload and exit the program
            Application.Exit();
        }
    }
}
```

Running the Program

After noting that the program is saved, press the F5 to run the Password Checker application. The Password Checker window will appear on the graphical display as shown in Figure 5.19. Notice the professional appearance and presentation of information in a clean dialogue box.

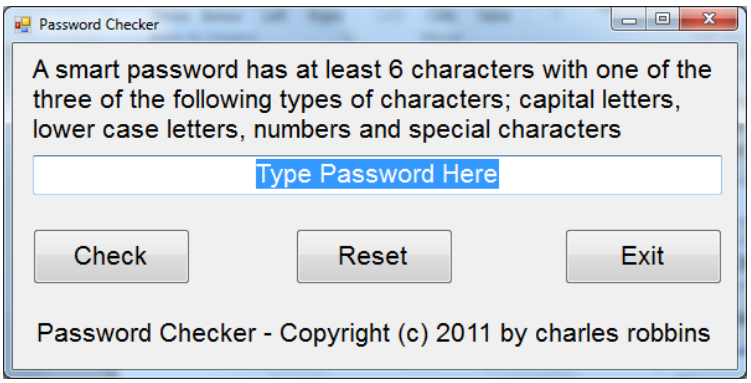

Figure 5.19 – Launching the Program

Type the password "A1B2C3d4" as shown in Figure 5.20. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Check command button and a message box will tell us if the password is strong and if it is not, how to fix the passphrase. After experimenting with our program, press the Exit command button to exit the application.

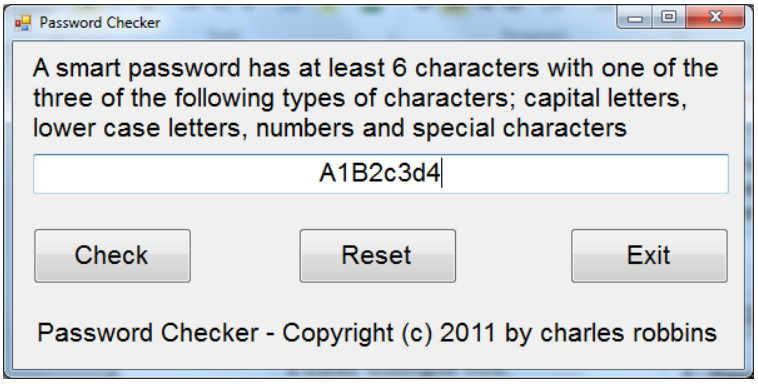

Figure 5.20 – Running the Program

When we check the password, we get the "Password is strong" message.

Figure 5.21 – A Message Box Appears

Now, we type the password "WorldClassCAD" as shown in Figure 5.22. Press the Check command button and a message box will tell us if the password is strong and if it is not, how to fix the passphrase.

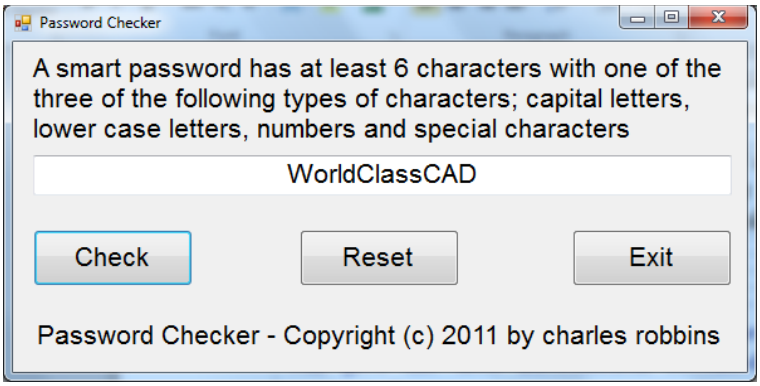

Figure 5.22 – Enter another Password

When we check the password, we get the "Password is not strong" message, that we need a number and a special character.

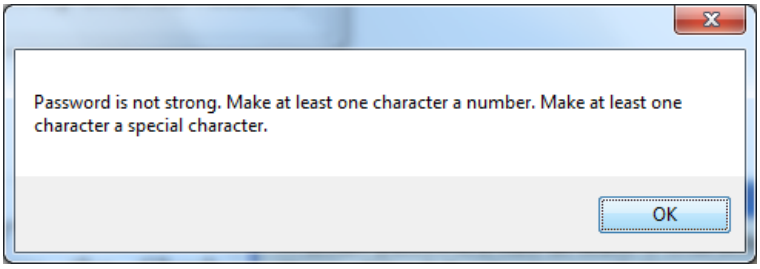

Figure 5.23 – A Message Box Appears

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 90-4 * - 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.